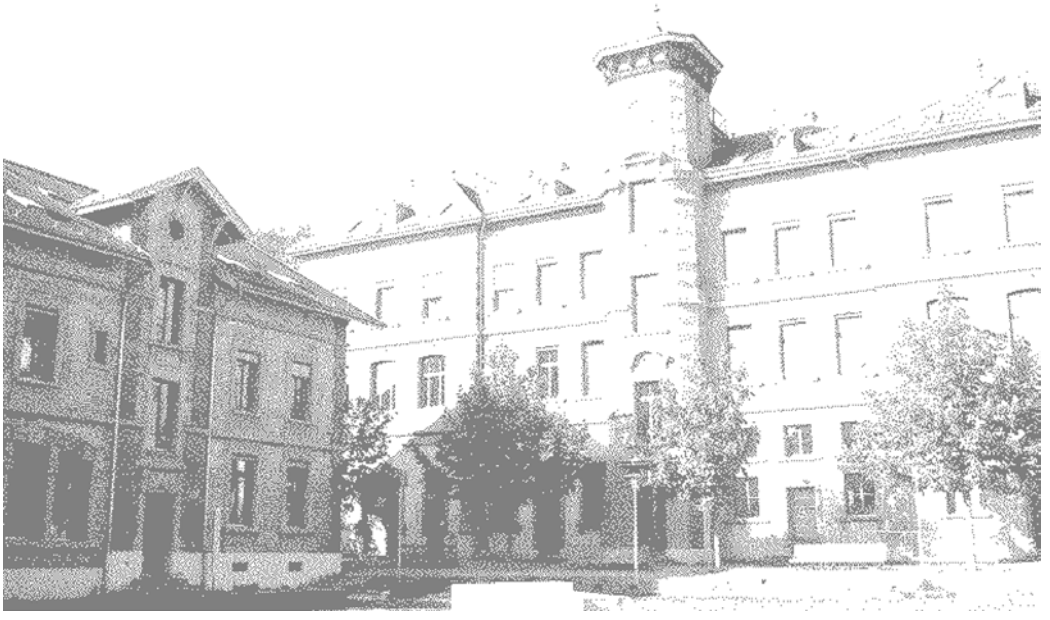
UNIVERSITÄT
BERN

2015 Doctoral Workshop on Distributed Systems

H. Mercier, T. Braun, P. Felber, P. Kropf, P. Kuonen, E. Riviere (eds.)

Technical Report INF-15-002, August 6, 2015

Institut für Informatik, www.inf.unibe.ch



Doctoral Workshop on Distributed Systems

Hugues Mercier, Torsten Braun, Pascal Felber, Peter Kropf, Pierre Kuonen, Etienne Riviere (eds.)

Technical Report INF-15-002, August 6, 2015

CR Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design; C.2.2 [Computer-Communication Networks]: Network Protocols; C.2.3 [Computer-Communication Networks]: Network Operations; C.2.4 [Computer-Communication Networks]: Distributed Systems

General Terms:
Design, Management, Measurement, Performance, Reliability, Security

Additional Key Words:
Wireless networks, information-centric networking, cloud computing, software transactional memory, fault tolerance, cellular networks, localization, wireless sensor networks, Internet of Things, privacy, security, energy efficiency, distributed systems, big data, distributed data storage

Institut für Informatik, Universität Bern

Abstract

The Doctoral Workshop on Distributed Systems has been held at Le Louverain, Neuchâtel, Switzerland, from June 21-23, 2015. Ph.D. students from the Universities of Neuchâtel and Bern as well as the University of Applied Sciences of Fribourg presented their current research work and discussed recent research results. This technical report includes the extended abstracts of the talks given during the workshop.

2015 Doctoral Workshop on Distributed Systems

Le Louverain
Neuchâtel
21 – 23 June

UNIVERSITÄT BERN

UNIVERSITÉ DE NEUCHÂTEL

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Workshop Program

Sunday, June 21

14:30 Welcome
Hugues Mercier and Etienne Rivière
14:40 Complex Systems and Big Data at UniNE
Valerio Schiavoni

Session 1 - System Integration

15:00 RESTful Semantics-Aware IoT Enterprise Integration
Matthias Thoma
15:40 Cloud-Based Computing-Oriented Service
Andrei Lapin

16:20 Coffee Break

Session 2 - Concurrency

16:50 Seamless Utilisation of GPU Resources for Runtime and Energy Reduction in the Actor Model
Yaroslav Hayduk
17:30 Performance Issues in Java GCs: Is HTM the Solution?
Maria Carpen-Amarie

Monday, June 22

Session 3 - Scaling and Performance

8:30 Service Level Agreements-Driven Management of Distributed Applications in Cloud Computing Environments
Alexandru-Florian Antonescu
9:10 Matching within the Order Book at the Stock Exchange
Raphaël Barazzutti

9:50 Coffee Break

Session 4 - Measurements

10:20 A Passive WiFi Localization System Based on Fine-Grained Power-Based Trilateration
Zan Li
11:00 Energy Efficiency in Heterogeneous Data Centers
Mascha Kurpicz
11:40 Deep Learning Feature Extraction for Natural Text Analysis
Baptiste Wicht

12:20 Lunch Break

Session 5 - Networking

14:00 On the Applications of Bloom Filters in Content-Centric Networks
Ali Marandi
14:40 NetCodCCN: a Network Coding Approach for Content-Centric Networking
Jonnahtan Saltarin

Tuesday, June 23

Session 6 - Routing

8:30 Information-Centric Wireless Communication
Carlos Anastasiades

9:10 Probabilistic Data Aggregation Protocol Based on Ant Colony Optimization in Wireless Sensor Networks
Yao Lu

9:50 Service Centric Networking
Bilal Gill

10:30 Coffee Break

Session 7 - Caching and Coding

11:00 Building a Trustworthy Long-Term Data Storage
Verónica Estrada Galiñanes

11:40 Cache Migration Strategies at the Edge of LTE Mobile Networks
André Gomes

12:20 Exploiting Caching to Improve the Performance of Geographically Distributed Stores
Raluca Halalai

Workshop Proceedings

Routing in Information-Centric Wireless Networks

Carlos Anastasiades
University of Bern
anastasiades@iam.unibe.ch

Abstract

Information-centric networking (ICN) offers new perspectives on wireless communication because messages are routed based on names and not endpoint identifiers. Since content can be uniquely identified by its name, requests can be dynamically forwarded to the closest content source. In this work, we consider ICN multi-hop routing in well-connected networks as well as ICN forwarding in delay-tolerant networks (DTN). While current routing protocols mostly consider only either of the cases, we show that ICN communication can be supported in both dense and sparse environments. We first describe *Dynamic Unicast (DU)*, an ICN routing protocol that builds unicast routes based on implicit content discovery. Then, we describe *Agent-based Content Retrieval (ACR)*, a DTN communication protocol, where content retrieval is delegated to agent nodes. All proposed communication protocols have been designed for Content-Centric Networking (CCN), which is a popular ICN architecture, and have been implemented in CCNx, the open source reference implementation of the CCN architecture.

Keywords: wireless multi-hop communication; delay-tolerant communication; information-centric networking.

1 Introduction

The shift from host-centric to information-centric networks (ICN) promises seamless communication in mobile and wireless networks because routing is based on names and not on endpoint identifiers. Since every content has a unique name and is signed, authentic content can be cached by any node. Instead of maintaining connectivity to specific hosts, users can exploit the broadcast nature of the wireless medium and retrieve content from the nearest content sources even if surrounding nodes may change.

In this work, we investigate wireless ICN routing in i) well-connected networks as well as ICN forwarding in ii) delay-tolerant networks (DTN). While current routing protocols mostly consider only either of the cases, we show that ICN communication can be supported in both dense and sparse environments. This is important because communication environments may not be strictly classified into one of the two groups. For example, network densities may change depending on location (urban vs. rural), daytime (rush hour vs. late at night) or season (summer vs. winter). Efficient network protocols should operate in isolated islands without connectivity to central infrastructures but should also benefit from these infrastructures if they are available. Thus, we assume that DTN communication may only be used for a certain time or at a certain place but eventually, users may use the infrastructure again to resume incomplete downloads or exchange collected information.

Our work is based on Content-Centric Networking (CCN) [1], which is a popular ICN architecture. In contrast to other ICN architectures, CCN requires no name resolution procedures, which makes it appealing for mobile and delay-tolerant networks (DTN). We have implemented Dynamic Unicast (DU) for information-centric wireless multi-hop communication and Agent-based Content Retrieval (ACR) for delay-tolerant communication. Since no modifications to CCN messages and processing are required to enable DTN communication, ACR can be combined with wireless multi-hop communication. For example, a requester can initially try to retrieve content via multi-hop communication and delegate content retrieval to other nodes (agents) if nothing has been found.

2 Work accomplished

2.1 Dynamic Unicast

Most wireless CCN approaches [2], [3], [4] ,[5] use broadcast transmissions exclusively to enable multi-hop communication. However, if all communication is performed via broadcast, all transmitted Data messages will be received and processed by all nodes in the sender's vicinity increasing the content density. As previous work [6] has shown, a high content density may require large forwarding delays to suppress duplicate Data transmissions. However, large forwarding delays reduce the achievable throughput for wireless communication. While forwarding delays are necessary for duplicate suppression during broadcast communication, they are not required during unicast transmissions. Therefore, we have implemented and evaluated *Dynamic Unicast*, which is based on flooding to find a content source and dynamic hop-by-hop configuration of unicast forwarding entries as Data returns on the reverse path, similar to Ad-hoc On-Demand Distance Vector Routing (AODV) [7].

However, there are three main differences to host-based routing protocols such as AODV. First, wireless CCN enables implicit content discovery in mobile networks, i.e., if a content source is available, it will reply Data to Interest messages. Thus, if con-

nectivity to a content source breaks, another content source can be found quickly (if available). Second, content can be cached in any node of a multi-hop path, thus, in case of collisions retransmissions do not need to be performed over the entire path. Third, Interests can be aggregated in each node such that fewer Interests are forwarded to a content source. Furthermore, Dynamic Unicast does not modify CCN messages by including node identifiers [2], [4] and does not require GPS coordinates [5] enabling interoperability with (partially) wired networks and deployments of resource constrained devices, where cache sizes and processing capabilities are very limited. We have implemented Dynamic Unicast with single path and multi-path forwarding strategy and compared it to multi-hop broadcast communication (default for wireless ICN communication).

Evaluations have shown that Dynamic Unicast with single path routing performs better than multi-path routing in terms of transmission times and message overhead. Furthermore, we have seen that CCN can effectively improve scalability of wireless communication because multiple requests can be aggregated and content can be retrieved from caches such that only a fraction of requests needs to be forwarded to content sources. Dynamic Unicast results in significantly shorter transmission times and fewer Data transmissions than broadcast for high content densities but surprisingly, it performes also better than broadcast for low content densities. This is because short contact times to content sources can be better exploited with Dynamic Unicast (no forwarding delays). Although broadcast communication is efficient to quickly find a content source, it is not required to perform all communication via broadcast.

2.2 Agent-based Content Retrieval

In delay-tolerant networks, existing protocols use device discovery [8], [9] to detect neighbor devices before communication takes place. However, the existence of neighbor devices does not reveal any information about available content or the neighbor's ability and willingness to perform certain tasks. Furthermore, if message structures or processing are modified as proposed in related work [10], [11], interoperability between dense and sparse environments would not be possible. Based on previous work [12], we have implemented and evaluated Agent-based content retrieval (ACR), where requesters can delegate content retrieval to other nodes (agents). Since DTN functionality is implemented as application module, it can also be combined with multi-hop communication in dense environments.

ACR is based on three phases. First, requesters need to find suitable agents in their vicinity and delegate content retrieval to them. This can be done via broadcast requests specifying content name and optional collection parameters, e.g., GPS coordinates where the content may be found, such that only agents that agree on the optional parameters (and can retrieve the content) may reply. Out of all replies, the requester can select the best agent and delegate content retrieval to it. Second, agents need to retrieve the content for the requesters. This can be performed via broadcast communication or Dynamic Unicast. Third, agents need to notify requesters that the content has been downloaded and can be retrieved from them. This can be done when requesters and agents meet the next time or via servers in the requesters' (or agents') home networks similar to custodian-based information sharing [13].

We have evaluated Agent-based Content Retrieval and Dynamic Unicast in diverse mobile scenarios. While Dynamic Unicast over multiple hops is faster in networks with high node densities, ACR is superior for low and intermediate node densities where multi-hop communication does not work or results in frequent disruptions. Furthermore, ACR is beneficial for large file sizes (lower message overhead, faster transmission times than Dynamic Unicast) and works well even under high mobility. Thus, node mobility is not necessarily a disadvantage for wireless communication and ICN provides the means to exploit it.

3 Work in progress and future work

The performance of *Dynamic Unicast* does not degrade with pedestrian mobility speeds. However, with vehicular speeds, transmission times increase significantly compared to static scenarios. The main reasons for the degradation are path breaks that are only detected after Interests have timed out. While the number of Interest transmission increases slightly for more mobile scenarios, the number of transmitted Data messages remains nearly constant. Thus, adaptive Interest lifetimes based on measured round-trip times are required to enable more seamless communication in high mobility scenarios. In addition, wireless CCN communication may be optimized by adapting pipeline sizes dynamically such that many Interests are transmitted on established paths but only a few Interests during implicit content discovery via broadcast.

The performance of *Agent-based Content Retrieval* depends on the mobility of agent nodes. To enable efficient ACR in arbitrary environments, new agent selection mechanisms need to be developed. Existing forwarder selection mechanisms in DTN networks are based on periodic neighbor discovery to detect communities. However, detected communities only show which devices have been in direct transmission range but do not indicate any interests or available content. Thus, wireless CCN should avoid periodic hello beacons to detect neighbor devices but exchange information only if required. To achieve this, new agent selection criterias need to be identified. Potential options

are for example GPS coordinates, content maps based on overheard or past Data transmissions, or social relations.

To increase the robustness of agent-based content retrieval, requesters can delegate content retrieval to multiple agents at the same time. However, multiple delegations may decrease communication efficiency due to redundant transmissions. Earlier work [14] showed that Raptor codes can result in improved communication efficiency for multiple concurrent requesters in wireless information-centric networks. Thus, if combined with ACR, Raptor codes may result in potential bandwidth savings because agents retrieve more diverse content and can benefit from each other's transmissions.

ACR can be combined with DU in different ways (hybrid approaches), which need to be further investigated. For example, a requester could initially try to retrieve content via multi-hop communication and only delegate content retrieval to agents if nothing has been found. Furthermore, agent delegations may also be performed over multiple hops. Because all messages are stored in the same format, requesters could even retrieve (via multiple hops) content from agents, which were delegated by other requesters.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Network Named Content", in *5th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Rome, Italy, December 2009

[2] M. Meisel, V. Pappas, and L. Zhang, "Ad hoc networking via named data", in *5th ACM workshop on Mobility in the evolving Internet Architecture (MobiArch)*, p. 3-8, Chicago, USA, September 2010

[3] L. Wang, A. Afanasyev, R. Kuntz, and R. Vuyyuru, "Rapid Traffic Information Dissemination using Named Data", in *1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms and Applications (NoM)*, pp. 7-12, Hilton Head Island, SC, USA, June 2012

[4] M. Amadeo, A. Molinaro, G. Ruggeri, "E-CHANET: Routing, Forwarding and Transport in Information-Centric multihop wireless networks", in *Computer Communications*, vol. 36, Issue 7, pp. 192-803, April 2013

[5] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida "Navigo: Interest Forwarding by Geolocations in Vehicular Named Data Networking", in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Boston, USA, June 2015

[6] C. Anastasiades, and T. Braun, "Dynamic transmission modes to support opportunistic information-centric networks", in *International Conference on Networked Systems (NetSys)*, pp. 1-5, Cottbus, Germany, March 2015

[7] C. E. Perkins, and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing", in *2nd IEEE workshop on Mbile Computing Systems and Applications (WMCSA)*, New Orleans, LA, USA, February 1999

[8] M. Pitkänen, and J. Ott "Mobility and service discovery in opportunistic networks," in *IEEE Pervasive Computing and Communications (PERCOM) workshops*, pp. 204-210, Lugano, Switzerland, March 2012

[9] M. Orlinksi, and N. Filer, "Neighbor discovery in opportunistic networks," in *Ad Hoc Networks*, Volume 25, Part B, pp. 383-392, February 2015

[10] F. Neves dos Santos, B. Ertl, C. Barakat, T. Spyropoulos, T. Turletti, "CEDO: Content-Centric Dissemination Algorithm for Delay-Tolerant Networks," in *16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 377-386, Barcelona, Spain, November 2013

[11] E. Monticelli, B. M. Schubert, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan, "An information centric approach for communications in disaster situations," in *20th IEEE International workshop on Local & Metropolitan Area Networks (LANMAN)*, pp. 1-6, Reno, NV, USA, May 2014

[12] C. Anastasiades, W. El Maudni El Alami, and T. Braun, "Agent-Based Content Retrieval for Opportunistic Content-Centric Networks," in *12th International Conference on Wired & Wireless Internet Communications (WWIC)*, pp. 175-188, Paris, France, May 2014

[13] V. Jacobson, et. al. "Custodian-based information sharing", in *IEEE Communications Magazine*, Volume 50, Issue 7, pp. 38-43, July 2012

[14] C. Anastasiades, N. Thomos, A. Striffeler, T. Braun, "RC-NDN: Raptor Codes Enabled Named Data Networking", in *IEEE International Conference on Communications (ICC)*, London, UK, June 2015

Service Level Agreements-Driven Management of Distributed Applications in Cloud Computing Environments

Alexandru-Florian Antonescu
University of Bern
antonescu@iam.unibe.ch

Abstract

Recent advancements in cloud computing have enabled the proliferation of distributed applications, which require orchestration and control of multiple services running inside virtualised containers deployed in network-connected cloud environments. However, without efficient mechanisms for scaling services in response to changing workload conditions, application performance might suffer, leading to violations of Service Level Agreement (SLA) and/or inefficient use of available resources. In this research work we tackle the complexity associated with scaling applications composed of multiple services by using a semantic model for representing distributed applications and their relations to key-performance indicators and SLAs as input for various autonomous control algorithms used for adjusting the horizontal scale of services. The key for achieving this is in extending the typical reactive approach to application scaling with benchmark-based pre-calculations of optimal load factors in relation to average response times, predicted arrival rate and average throughput. We are describing such a scalability control algorithm and its evaluation in a real-world scenario involving a distributed enterprise information system (dEIS). We will present the results of a large-scale simulation, where the dEIS workload was increased by four orders of magnitude, from 5 to 25'000 simulated users, while the control system had to maintain an average response time of below one second. Our enhanced management algorithm managed to maintain 100% SLA compliance by appropriately varying the number of allocated VMs, as opposed to a similar non-calibrated reactive scaling algorithm. These results were selected from the PhD thesis which will be defended on 2015 October 23rd.

Keywords: cloud computing, service scaling, SLAs.

1 Introduction

Recent advancements in cloud computing have enabled the proliferation of distributed applications, which require orchestration and control of multiple services running on virtualised containers inside network-connected cloud environments. However, without an efficient mechanism for scaling services in response to changing workload conditions, application performance might suffer, leading to Service Level Agreement (SLA) violations and/or inefficient use of hardware resources. Combining dynamic application requirements with the increased use of virtualised computing resources creates a highly challenging resource management situation for application and network and computing infrastructure owners.

In such complex environments, business entities use SLAs as a means for specifying quantitative and qualitative requirements of services. Given that SLAs have legal and financial implications, proper management of SLAs is critical for modern and future application and infrastructure service providers. Leading cloud service providers of infrastructure or application services (e.g. Google, Amazon) use SLA management for specifying and maintaining the quality of service (QoS) and availability levels to their customers. Dealing with exclusively-owned virtual machine (VM) instances deployed on a shared physical infrastructure presents a bigger challenge for each resource management lifecycle phase, given (1) the multi-objective resource allocation optimization problem of having to maintain SLAs for various customers while minimising resource utilisation targets, and (2) the differentiation in SLA-requirements from different classes of VMs and VM users. Furthermore, the violation of SLAs results in cash penalties for the cloud-infrastructure provider, adding a direct economic dimension to the problem.

We introduce the requirements and the architecture of a SLA-aware cloud management system for controlling the complexity of scaling applications composed of multiple services using mechanisms based on fulfilment of SLAs. For this, we provide the answers to the following research questions:

1. "How do distributed application owners and infrastructure providers benefit from using Service Level Agreements in Cloud Computing environments?"
2. "How to control infrastructure resource allocation in cloud environments while maximising both user QoS and provider Efficiency of Operations?"
3. "How to design an effective system for management (scaling) of SLA-constrained distributed cloud services?"

In order to answer these questions, we considered multiple connected problems from the perspec-

tive of both enterprise application owners and cloud infrastructure providers. The efficient management of SLAs is of particular importance for Cloud Computing, where exclusively-owned Virtual Machines (VMs) are allocated resources on hosts in a shared physical infrastructure.

The application owner is interested in finding the optimum amount of computing and network resources to ensure that the performance requirements of all applications are met. After the initial allocation of virtual computing resources, the application owner is also interested in appropriately scaling distributed applications so that application performance guarantees are maintained even under dynamic user-generated workload conditions.

Similarly, the infrastructure providers are interested in optimally allocating the virtual resources on the physical infrastructure resources so that its operational costs are minimised, while maximising the tenant's application performance. In this context, we show how predicting the infrastructure utilisation peaks and dynamically optimising the distribution of virtual resources is a viable approach for optimising the use of available cloud resources, in comparison to simple or reactive (i.e. non-predictive) approaches.

2 Main Contributions

Overall, the key contributions of this dissertation[1] are:

1. a semantic SLA-enabled specification language and architecture for dynamically managing distributed software and computing, storage and network infrastructure services [1]

2. a framework for dynamically allocating VMs to physical resources while considering SLA-constraints and multiple objectives optimisations by using a genetic algorithm combined with data forecasting (exponential smoothing) [2]

3. a method of sizing virtual infrastructures based of SLA-defined constraints [3], and composing scaling rules for distributed services using prediction mechanisms and correlation-derived relationships between SLA monitoring metrics [4]

4. two SLA-based VM scaling algorithms that use reactive and analytic mechanisms combined with data prediction methods and results from applying Little's Law ([6, 7, 8])

5. implementation of the previous mentioned techniques and tools for SLA-aware management of cloud infrastructure and software resources in the Service Middleware Layer (SML) component of the GEYSERS FP7 EU research project ([10], [11], [12])

[1] `http://www.iam.unibe.ch/~antonescu/phd.pdf`

We define the research question as: *"How can a Cloud Management System (CMS) dynamically scale the number of VMs allocated to cloud services, so that the SLA-defined performance constraints are maintained under variable workload conditions such as fluctuating number of users?"*.

2.1 Semantic SLA-Enabled Framework for Dynamic Management of Distributed Applications

In order for a CMS to be able to autonomously manage virtual infrastructures composed of services running across multiple VMs, it must be provided with a machine-readable representation of the application services, their requirements in terms of cloud resources, the relations between them, and the associated SLA monitoring metrics. For satisfying these conditions, we describe a semantic model for representing application topologies combined with SLA conditions and guaranteed CMS-actions [1].

The model used for defining the SLAs in relation to the monitoring metric associated with the application services was built using RDF [13] by extending the USDL-SLA [14] vocabulary. It defines SLA guaranteed states and actions (management rules), as well as the conditions required for automatic execution of the actions, which enable the CMS to autonomously take action on certain conditions potentially leading to SLA violations.

Once the semantic SLA model has been created by application topology architects, it will be given as input to the CMS [1]. The CMS will perform the dynamic topology orchestration process, by performing multiple stages. (1) It will analyse the services contained in the SLA, the relations between them and their resource requirements, followed by the scheduling of appropriate infrastructure operations. (2) The cloud infrastructure will be prepared by reserving the corresponding amount of resources (e.g. network, storage, computing). (3) Application services will be deployed, followed by activation of probes for monitoring the services' performance. (4) The distributed application will be monitored, with the data being fed to the CMS for evaluating the SLA guarantees and action triggers. (5) The CMS executes the SLA-defined operations for ensuring that the distributed system remains in the SLA-defined performance boundaries. Possible CMS-operations are: scale-out - adding VM instances; scale-in - terminating VM instances, scale up - increasing resources (e.g. CPU, memory) allocated to a VM; scale-down - decreasing the amount of resources allocated to a VM.

2.2 Multi-Objective SLA-Aware Allocator of VMs in Cloud Environments

It is often the case that at datacenter level the CMS must solve a complex multi-objective optimization resource allocation problem. The physical infrastructure providers must deliver and maximise the SLA-advertised levels of performance and capacity availability, while minimising energy consumption, resource wastage and costs created by violating the agreed SLAs.

In order to solve this problem, we described a CMS architecture using a VM allocator based on a multi-objective genetic algorithm, and predictive mechanisms [2]. The system allocates VMs to physical hosts while considering the effect (e.g. penalties) of the existing SLAs and using historical monitoring data to forecast the incoming load on both the physical and virtual infrastructure, in order to (1) maximise SLA satisfaction, (2) minimise penalties, and (3) minimise energy consumption at datacenter level.

Given that the problem on allocating VMs to physical hosts is equivalent to the NP-hard bin-packing problem, we decided on using a genetic algorithm (GA) [15] for solving it. Also, given the fact that the VMs are naturally grouped by their hosts, we selected a group-oriented GA [16] for finding a solution to the VMs allocation problem.

The proposed GA works by encoding VMs as genes, hosts as groups and the entire datacenter as a single chromosome. The genetic operator *mutation* works by randomly deleting a gene (VM), followed by its reinsertion using a heuristic algorithm (e.g. first-fit), ensuring that no VMs are simply eliminated. The genetic *crossover* operator works by selecting from each of the two chromosomes the groups with the highest fitness value. As this might lead of omission of some VMs from the resulting chromosome, they will be reinserted using the same heuristics as in the case of the mutation operator.

For calculating the fitness of a chromosome we considered four objectives in for the allocation of VMs: maximising the total revenues, minimising the datacenter's energy costs, minimising the costs associated with VM migrations, and minimising SLA penalty costs. These different objectives are combined in an aggregate objective function by assigning weights to each of the four mentioned costs.

2.3 SLA-Based Virtual Infrastructure Sizing and Dynamic Composition of VM-Scaling Rules

Dynamically changing (scale out/in) the number of VMs allocated to a distributed services is one of the main mechanisms used in cloud environments for adapting the performance of applications to variation in workload conditions. In [3] we proposed using a dynamically generated scaling model for the VMs associated to services of distributed applications, for enabling a CMS to scale VMs as a reaction to variations in the number of application users, so that it can maintain SLA-defined performance guarantees. We answered the following research question: "How to dynamically decide how many services instances are needed in order to handle a larger workload within the same time constraints?".

The Dynamic SLA Optimiser (DynSLAOp) system, which receives as input a semantic-SLA specification for a distributed application, together with an upper limit for the application's performance. DynSLAOp instantiates the required cloud resources (VMs, network connections) and then it begins benchmarking the distributed application with an increasing number of application users until the SLA-defined performance bound is reached. The monitoring information about service instances is recorded for further processing. After completing the current benchmark, the number of VMs is increased and the benchmark is repeated. This process is performed until the user-defined maximum number of VMs is reached.

After all benchmarks have been executed, DynSLAOp creates a scaling "path" (ScP) for supporting a monotonically increasing number of application users. One ScP is a collection of tuples $(n_{S_1}, n_{S_2}, ..., u)$, where n_{Sx} is the number of VM instances allocated to service S_x and u is the number of application users. The monitoring information corresponding to each ScP-tuple is then consolidated into a large dataset, with only the data entries corresponding to the selected number of users (u). The dataset is then analysed for determining correlations between the critical performance metric (given to the system) and the application's monitoring SLA-metrics. The set of correlated parameters is then used for building a multi-variable regression model (MVRM) for estimating the number of VMs (n_{S_x}) of each service (S_x) in ScP [3].

The MVEM is finally converted into a SLA scaling rule. The CMS continuously compares the MVEM's produced value with the actual number of VMs in the system and performs either scale-out or scale-in in order to keep the number of VMs synchronised.

2.4 Predictive SLA-Aware Scaling of Distributed Services in Cloud Environments

In order to validate our assumptions related to SLA-constrained VM-scaling of distributed cloud applications we set to perform a large-scale simulation of an enterprise application. For this purpose, we first described a method of profiling a distributed application and building a simulation model for it [7]. We did this by first deploying exactly one VM instance of each service of the distributed application, followed by running multiple benchmarks with a constant number of

application users. This allowed us to record the CPU, memory, network I/O, storage I/O utilisation at the same time with application level metrics, under constant workload conditions. All the monitoring datasets were then consolidated in a large dataset, where each entry represented a distributed transaction across the enterprise application, tagged with the number of concurrent transactions executed by the system.

In the next step we converted this consolidated dataset into a simulation model by converting the measured execution durations into MIPS (million instructions per second) units. This allowed us to adapt the consolidated model of the enterprise application to CloudSim [17] simulator. Our validation experiments and analytical testing (using Kolmogorov-Smirnov test and the linear dependency coefficient) showed that the simulation model accurately modelled the behaviour of the real enterprise application.

Having an accurate model of a real enterprise distributed application, we tested whether a reactive SLA-based VM-scaling algorithm could be used for managing our distributed enterprise application [6]. We also evaluated the ability of CloudSim to simulate multiple cloud tenants. We have successfully simulated two cloud tenants with and without VM-scaling with a dynamic, slowly changing, workload. This validated the use of CloudSim for multi-tenant, large-scale cloud simulations.

We described two novel SLA-based VM-scaling algorithms, which use the SLA-defined maximum value for the average execution time for determining the maximum processing throughput of each service [8]. The optimum throughput (Th_{max}) is calculated by applying Little's Law [18] to the consolidated dataset described in [7]. The first VM-scaling algorithm works by assuming that the system operates at equilibrium, where the system's throughput (Th) is (almost) equal to the workload's rate of arrival (λ). The VM-scaling algorithm then calculates the number of VMs (vm^*) that the system should have so that under uniform load-balancing, each VM will process at most 80% of the Th_{max}. If wm^* is higher than the current number of running VMs (vm) plus the number of VMs created by not yet started (vm^+), then the CMS will create an additional number of VMs, equal to $vm^* - vm - vm^+$.

The second VM-scaling algorithm from [8] built on the previously described algorithm by using prediction to forecast the workload's rate of arrival (λ), and then it used the predicted-λ for determining the expected number of VMs for a time horizon of a few minutes. Knowing the average VM's instantiation duration allowed the CMS to anticipate the VMs-scale-out operations and to execute the scale-out before the actual scaling conditions were met, ensuring that the distributed system was always in a SLA-compliant state.

3 Evaluation

We briefly introduce the application under test, which we used throughput ([1], [3], [4], [7], [8], [12], [9], [5]). It is composed of four core services composing the distributed enterprise application (dEIS): one or more Thin Clients (CS), a Load Balancer (LB), one or more Worker services (WK), and one or more Database Storage services (ST). Each service runs in its own VM and communicates asynchronously with the other services using a distributed service messaging bus. The dEIS's components are presented in [1], and dEIS's performance characteristics are described in detail in [7].

The SLA semantic model [1] described in Section 2.1 was used in the implementation of the Service Middleware Layer (SML) cloud infrastructure management platform ([10], [11], [12], [19]). We evaluated the model's capacity to represent complex service level objectives (SLOs), as well as the relations between cloud services, and the conditions for CMS-automated execution of cloud-infrastructure operations.

The multi-objective VM-allocator algorithm described in Section 2.2 was evaluated by means of a large-scale simulation, in which we considered the impact of VM migrations and CPU oversubscription on VM allocation operations. Our results [2] have shown that using a multi-objective VM allocator based on a group-oriented genetic algorithm, combined with resource-utilisation prediction mechanisms can lower the costs associated with CPU and network oversubscription, when compared with well-known heuristics-based allocation algorithms.

The methods and algorithms for dynamic composition of service-scaling rules described in Section 2.3 were evaluated by scaling the services of dEIS system as described in [3]. These experiments enabled creating an analytic model of the dEIS's performance, which we then used for the large-scale simulation described in Section 2.4.

We evaluated VM-scaling algorithms from Section 2.4 with a real-world workload (described in [8]), which varied the number of users from 1 to 25'000 during approximately four hours. The CMS created approximately 150 VMs, validating the stability and effectiveness of the VM-scaling algorithms, as they were able to maintain the average execution time below the maximum value defined in the SLA.

4 Conclusions

The main contribution of this dissertation is on the management of scalable distributed enterprise applications. We summarise as follows the researched set of novel techniques and tools for datacenter-infrastructure resource allocation, VM-scaling, and management of cloud-services.

For cloud-deployed multi-tier applications it is often the case that the number of allocated infrastructure resources have to be dynamically changed. This

dissertation proposes a novel semantic SLA model for supporting (1) the description of complex, interconnected distributed applications, their monitoring metrics, infrastructure management operation, and SLOs; (2) the automated execution of infrastructure-scaling operations based on SLA-defined triggers.

Convergent management of cloud infrastructure resources (network, computing, storage) requires considering multiple objectives, such as (1) minimising the costs associated with violating SLAs due to oversubscription of network and computing resources, (2) maximising the tenants' quality of service, and (3) minimising energy consumption. This dissertation proposes a novel resource allocation algorithm for virtual machines based on a genetic algorithm, which combines uses prediction mechanisms for optimally allocating VMs, while satisfying multiple objectives concerning the efficiency of the management operations.

Multi-tier distributed applications have complex inter-dependencies between their services, making design of scaling rules a complicated process. This dissertation proposes using benchmarks for exploring the dependencies between the number of allocates services of a distributed application and its maximum processing capacity. We present (1) a method of using correlations for discovering the key parameters that determine the number of required service instances of a distributed application; and (2) an algorithm for composing SLA-based VM-scaling rules based on the benchmark-obtained datasets.

Different cloud applications have varied requirements regarding the value of quality of service metrics, making automated scaling of such systems a complicated operation. This dissertation proposes different VM-scaling algorithms, which are able to incorporate SLA-derived performance requirements for (1) finding the maximum per-VM processing capacity of each distributed service, (2) and then use this for optimally-scaling the number of VMs such that, under varying workload conditions, the level of performance experienced by the application's users remains in the bounds defined by the SLAs.

References

[1] A.-F. Antonescu, P. Robinson, and T. Braun, "Dynamic topology orchestration for distributed cloud-based applications," in *Proc. 2nd IEEE Symposium on Network Cloud Computing and Applications (NCCA)*, 2012.

[2] A.-F. Antonescu, P. Robinson, and T. Braun, "Dynamic SLA management with forecasting using multi-objective optimizations," in *Proc. 13th IFIP/IEEE Symposium on Integrated Network Management (IM)*, May 2013.

[3] A.-F. Antonescu, A.-M. Oprescu *et al.*, "Dynamic optimization of SLA-based services scaling rules," in *Proc. 5th IEEE Internetional Conference on Cloud Computing Technology and Science (CloudCom)*, December 2013.

[4] A.-F. Antonescu and T. Braun, "Improving management of distributed services using correlations and predictions in SLA-driven cloud computing systems," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, May 2014.

[5] A.-F- Antonescu and A. Gomes and P. Robinson and T. Braun, "SLA-driven predictive orchestration for distributed cloud-based mobile services". in: Kim, Dong-In; Mueller, Peter (eds.) 2013 *IEEE International Conference on Communications Workshops (ICC)* (pp. 738-743). IEEE 10.1109/ICCW.2013.6649331

[6] A.-F. Antonescu and T. Braun, "SLA-driven simulation of multi-tenant scalable cloud-distributed enterprise information systems," in *ACM PODC Workshop on Adaptive Resource Management and Scheduling for Cloud Computing (ARMS-CC)*, 2014.

[7] A.-F. Antonescu and T. Braun, "Modeling and simulation of concurrent workload processing in cloud-distributed enterprise information systems," in *ACM SIGCOMM Workshop on Distributed Cloud Computing (DCC 2014)*, August 2014.

[8] A.-F. Antonescu and T. Braun, "Simulation of SLA-based VM-scaling algorithms for cloud-distributed applications," *Future Generation Computer Systems (FGCS)*, 2015.

[9] A.-F. Antonescu and T. Braun, "Service Level Agreements-Driven Management of Distributed Applications in Cloud Computing Environments" in *IFIP/IEEE* Symposium on Integrated Network Management (IM 2015), Ottawa, Canada, May 11-15 2015

[10] A.-F. Antonescu and P. Robinson, "Towards cross stratum SLA management with the geysers architecture," in *Proc. 10th IEEE Int. Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2012.

[11] E. Escalona *et al.*, "Geysers: A novel architecture for virtualization and co-provisioning of dynamic optical networks and it services," in *Future Network & Mobile Summit (FutureNetw)*. IEEE, 2011, pp. 1–8.

[12] J. F. Riera, J. Garcia-Espin, A.-F. Antonescu *et al.*, "Virtual infrastructures as a service enabling converged optical networks and data centres," *Optical Switching and Networking (OSN)*, pp. 197–208, 2014.

[13] G. Klyne and J. J. Carroll, "Resource description framework (RDF): Concepts and

abstract syntax," http://www.w3.org/TR/2004/REC-rdf-concepts-20040210, 2004.

[14] Leidig, T. and C. Momm, "USDL service level agreement," http://www.linked-usdl.org/ns/usdl-sla, April 2012.

[15] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA, 1996.

[16] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, no. 1, p. 5,Äì30, 1996.

[17] G. Lab, "Cloud simulator cloudsim," http://code.google.com/p/cloudsim, 2014.

[18] J. D. Little and S. C. Graves, "Little's law," in *Building Intuition: Insights From Basic Operations Management Models and Principles*. Springer, 2008, pp. 81–100.

[19] A.-F. Antonescu, P. Robinson, and M. Thoma, "Service level management convergence for future network enterprise platforms," in *Future Network & Mobile Summit (FutureNetw)*, 2012, pp. 1–9.

Matching within the order book at the stock exchange

Raphaël P. Barazzutti
Université de Neuchâtel
raphael.barazzutti@unine.ch

Abstract

Concurrent programming is essential to exploit parallel processing capabilities of modern multi-core CPUs. While there exist many languages and tools to simplify the development of concurrent programs, they are not readily applicable to domain-specific problems that rely on complex shared data structures associated with various semantics (e.g., priorities or consistency).

In this research, we focus on a application from the financial domain, where a data structure named *order book* is used to represent a double auction market. This structure store and match orders from buyers and seller arriving at a high rate. This application has interesting characteristics as it exhibits some clear potential for parallelism, but at the same time it is relatively complex and must meet some strict QoS guarantees, notably respect the ordering of operations. We first present an accurate yet slightly simplified description of the order book problem and describe the challenges in parallelizing it. We then introduce several approaches for introducing concurrency in the shared data structure, in increasing order of sophistication starting from lock-based techniques to lock-free designs. We propose a comprehensive workload generator for constructing histories of orders according to realistic models from the financial domain.

Keywords: concurrency; orderbook; stockexchange; high frequency trading.

1 Introduction

Nowadays, most stock exchanges provide a fully automated order matching platform to its participants. For each stock exchange security available on the market, a broker maintains a structure called an "order book" (see Figure 1), which agglomerates a set of orders received from clients. To facilitate order matching, a matching engine is used. Since a large number of participants are doing high frequency trading, it is essential for the matching platform of the stock exchange to provide its services at a very low latency.

As brokers and traders are already expecting the latencies to be in the order of a few milliseconds, the stock exchange needs to have internal latencies that are at least one order of magnitude lower. To facilitate these low latencies, brokers started looking into new communication mechanisms [1] to gain a few millisecond advantage. Recently, some brokers reached ultra-low latencies, i.e., SIX was the first one in 2012 to have reached latencies under $50\mu s$[2]. This time accounts

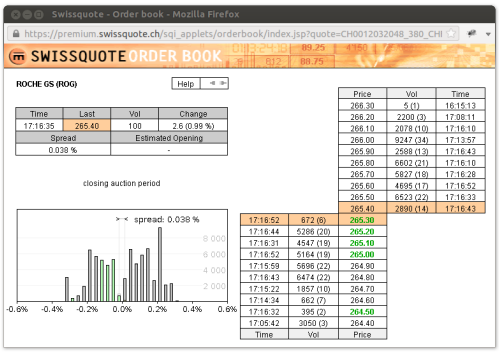


Figure 1: An order book, as seen on Swissquote

for the full roundtrip to the matching engine and back, including validation, processing and acknowledgement of the participant order.

Instead of concentrating on communication mechanisms, in this paper we are targeting to improve the effectiveness of the matching engine in order to improve the service latency. Current matching engines thus far work sequentially, which means that, despite the system capacity to receive multiple orders concurrently, the processing of orders is handled in sequence. As there is a great potential of obtaining performance benefits from exploiting parallel processing capabilities of modern multi-core CPUs, we are targeting to experiment and compare a number of strategies for concurrent order processing. Currently there exist many languages and tools to simplify the development of concurrent programs, however they are not readily applicable to domain-specific problems, such as the one we are describing in this paper.

Order matching has interesting characteristics as it exhibits some clear potential for parallelism. At the same time it is not trivial, and presents a number of challanges which need to be addressed. First, we need to ensure that the output of the matching process in the concurrent case is the same as the sequential case. Also, as in the sequential case, the concurrent implementation must meet some strict QoS guarantees. Notably, one of the main challenges in processing orders concurrently is in that we need to process incoming orders in the order in which they were initially received (FIFO order). In addition, as the system handles a number of messages types (add/remove, sell/buy), it is not clear whether some combinations of messages can be handled safely when processed concurrently (i.e., the order book data structure is not thread-safe). Lastly, to fulfill an order, the matching

engine can potentially access more then one order already stored in the order book. In the concurrent case this might lead to many message processings accessing the same shared state; here, special precautions need to be taken to avoid possible data corruption associated with concurrency hazards.

2 Work accomplished

2.1 Workload generators

In order to generate workloads having similar statistical properties to the real world workloads, we carefully selected state-of-the-art generation mechanisms used in economics and econophysics. We implemented an algorithm described by Maslov[5], which is well-known in the econophysics community and is very straightforward.

Succinctly, the orders to be submitted are pregenerated with a set of generation rules; a random number generator for creating order insertions and order removals is used. Order removals are used to purge limit orders, which have stayed in the order book for more than a predefined number of time steps (one thousand in Maslov's paper[REF?]). To support this generation algorithm, a full-fledged matcher is required to properly handle order removals.

We also implemented another generator using mechanisms proposed by Bartolozzi[6].

2.2 Matching engines

We started by implementing a non-concurrent matching engine, which we use as a baseline when comparing to concurrent alternatives. Then we developed two other implementations, one with a single lock mechanism, and one with fine-grained locking mechanisms.

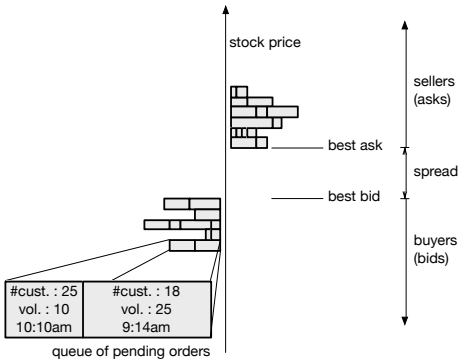


Figure 2: Internal structure of order book

Input: *order*
oppositeBook ← select an opposite order book based on *order.type*;
orderLists ← create an empty list;
remainingVolume ← *order.volume*;
foreach *orderList in oppositeBook.orderLists* **do**
  if *remainingVolume* > 0 **then**
    break;
  remainingVolume ← *remainingVolume* − *orderList.totalVolume*;
  orderLists.add(*orderLists*);
end
foreach *orderList in orderLists* **do**
  foreach *orderOpp in orderList* **do**
    perform matching;
    send *matchMessage* to *orderOpp.participant*;
    send *matchMessage* to *order.participant*;
  end
  //were all orders consumed from *orderList*?;
  if *orderList* ***is*** *empty* **then**
    remove *orderList* from *oppositeBook*();
end

Algorithm 1: Sequential order matching: Insertion

Input: *order*
orderBook ← select an order book based on *order.type*;
orderList ← from *orderBook* select *orderList* corresponding to *order.price*;
remove *order* from *orderList*;

Algorithm 2: Sequential order matching: Removal

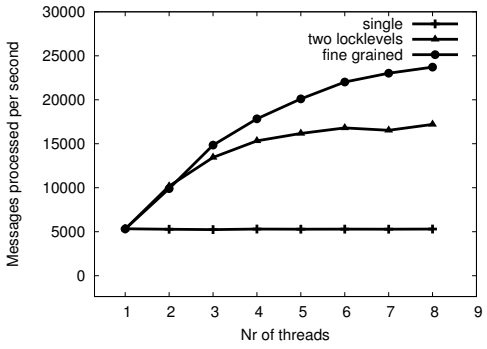


Figure 3: Evaluation of algorithms on an Intel i7

3 Work in progress and future work

Currently we are working on alternative lock-free implementations, which do not fully follow the semantics of a real-world matching engine (e.g., might not respect FIFO order processing), but might provide more insight on the amount of concurrency that we can obtain. Secondly, we would also like to test a version taking advantage of hardware transactions, which is available on the last generation of Intel CPUs (e.g., Broadwell) as well as on IBM's Power8.

References

[1] WIRED, "Raging Bulls: How Wall Street Got Addicted to Light-Speed Trading", available at `http://www.wired.com/2012/08/ff_wallstreet_trading/2/`

[2] CME Group, "What Happened on May 6th?", 2010, available at `http://www.scribd.com/doc/31546905/CME-Group-Report-on-the-Flash`

[3] SIX Swiss Exchange Ltd, "SIX - Directive 3: Trading", available at `http://www.six-swiss-exchange.com/rule_book/04-DIR03_en.pdf`

[4] SIX Swiss Exchange Ltd, "SIX - Directive 4: Market Control", available at `http://www.six-swiss-exchange.com/rule_book/05-DIR04_en.pdf`

[5] Maslov S., "Simple model of a limit order-driven market", *Physica A: Statistical Mechanics and its Applications*, 200, p. 571-578

[6] Bartolozzi M., "A Multi Agent Model for the Limit Order Book Dynamics", 2009

Performance issues in Java GCs: is HTM the solution?

Maria Carpen-Amarie
Université de Neuchâtel
maria.carpen-amarie@unine.ch

Abstract

In the last few years, managed runtime environments such as the Java Virtual Machine (JVM) are increasingly used on large-scale multicore servers. The garbage collector (GC) represents a critical component of the JVM and has a significant influence on the overall performance and efficiency of the running application. We first perform an extensive study on all available Java GCs, both in an academic environment (set of benchmarks), as well as in a simulated real-life situation (client-server application). We report considerable application pauses, which makes current GCs unfit for environments where responsiveness is crucial, such as realtime or distributed systems. There have been several approaches for developing concurrent GCs that can exploit the processing capabilities of multi-core architectures, but at the expense of a synchronization overhead between the application and the collector. We investigate a novel approach to implementing pauseless moving garbage collection using hardware transactional memory (HTM). We study the overheads resulting from using transactional barriers in the Java virtual machine (JVM) and discuss various optimizations. Our findings show that, while the cost of these barriers can be minimized by carefully restricting them to volatile accesses when executing within the interpreter, the actual performance degradation becomes unacceptably high with the just-in-time compiler. The results tend to indicate that current HTM mechanisms cannot be readily used to implement a pauseless GC in Java that can compete with state-of-the-art concurrent GCs.

Keywords: Garbage Collection; Hardware Transactional Memory.

1 Introduction

Today, the most often used server applications, such as Cassandra or JBoss, are executed on managed runtime environments like Java VM. An essential component of any managed runtime is the *garbage collector* (GC). The GC is used for automatically allocating and reclaiming memory, thus avoiding all memory leaks or wrong memory accesses. Nowadays all servers are multicore, with the number of cores still expected to increase in the following years. As such, a simple serial GC is not adequate any more. However, even though the most recent Java GC algorithms are parallel (i.e., multiple GC threads working at the same time) and/or concurrent (i.e., the GC threads work concurrently with the application threads), their performance still affects the application throughput or responsiveness [3, 2]. We try to overcome the issue by employing a lock-free mechanism, i.e., *transactional memory* (TM) [1], instead of the current blocking phases. Hardware TM (HTM), recently proposed in consumer-grade processors, appears to be the perfect solution for improving the performance of specialized concurrency problems, such as concurrent garbage collection.

We build on top of Java's *ConcurrentMarkSweep* GC. As mentioned before, in a concurrent collector, the application threads perform at the same time as the GC threads. Thus, they need to be synchronized with access barriers, which guard against concurrency hazards. We replace all the barriers (read and write) with transactions. We are trying to find if using HTM in GCs is a viable option. That is, if a pauseless HTM algorithm also has an acceptable throughput and could be readily used in a real-life scenario. Therefore, we mainly focus on evaluating the impact of the transactional barriers.

We tested our modified version of ConcurrentMarkSweep with the DaCapo benchmark suite on a 8-core Intel Haswell machine, with fully integrated HTM support. When implemented and optimized in the interpreter, we observe a reasonable throughput of around 4%. However, the equivalent overhead observed with the *just-in-time* compiler (JIT) is rather large, going up to extreme values, such as 120%. The results show that, in terms of throughput, our GC algorithm would not be able to compete with the existing Java concurrent GCs.

2 Work accomplished

Our contributions are as follows:

- **Extensive analysis on GC impact.** It represented our strong motivation for improving the existing concurrent GCs with HTM.
- **Transactional algorithm.** We devised a GC algorithm that tries to take advantage of HTM and implemented the transactional barriers.
- **Evaluation of transactional barriers.** We thoroughly tested our transactional barriers in order to find out if the potential pauseless GC can also maintain a comparable throughput level.

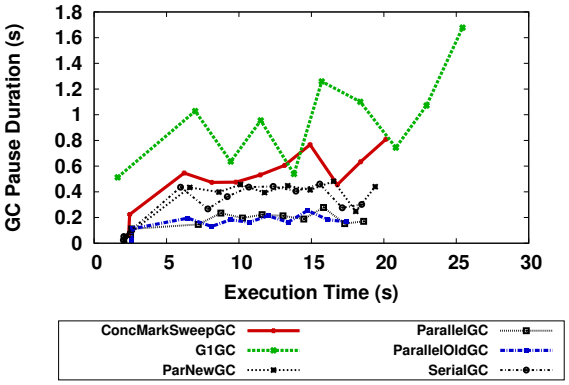


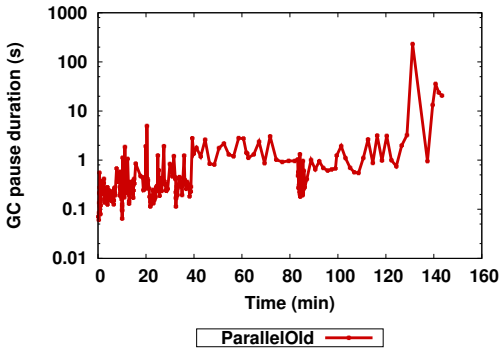
Figure 4: GC pauses for the benchmark scenario: Xalan benchmark

2.1 GC performance study

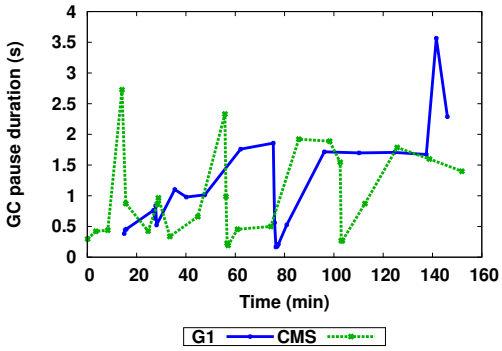
We studied the impact of garbage collection on application performance in two scenarios: on a benchmark suite and a client-server system. All tests were run on a 48-core server with 64 GB RAM. In this paper we focus on the application pauses caused by the collections (also called *stop-the-world* pauses). They represent our main motivation for developing a pauseless garbage collector. We are especially looking at the three most used Java GCs: ParallelOld (the default), ConcurrentMarkSweep (CMS) and Garbage-First (G1). All three GCs are also generational: all new objects are allocated in a part of the heap called the *young generation*. They are expected to die soon and only the objects that survive several young generation collections are promoted to the *old generation*. Typically, the young generation is collected more often and significantly faster than the old generation. An old generation collection is also called *full GC*.

For the **benchmark scenario**, we selected a subset containing all DaCapo benchmarks [5], that were stable in execution time. Each benchmark had ten iterations and a system (full) GC between any two iterations. We tested with all Java GCs, varying at the same time the heap size, young generation size and other properties. We considered as baseline the default Java configuration on the testing machine: 16 GB heap size, 6 GB young generation size. Figure 4 shows the pauses caused by the GCs (on the Y axis) and the execution time (on the X axis) for the Xalan benchmark. Please note that the results are similar for the other benchmarks. The highlighted curves represent the three main Java GCs on which we are focusing our analysis. We observe that the default Java GC, ParallelOld, performs better in terms of both pauses and throughput (given by the execution time) than the concurrent GCs (i.e., G1 and ConcurrentMarkSweep).

However, when testing on a real-life scenario, on the client-server system, the performance of the GCs significantly changes. We test with the Apache Cassandra database server [6] and the YCSB benchmark [7], on



(a) ParallelOld



(b) CMS & G1

Figure 5: GC pauses for the client-server scenario

the baseline configuration. Figure 5 shows the pauses caused by the GC activity in two different charts, separately for ParallelOld and for CMS and G1, respectively. In the case of ParallelOld, the pauses are with two orders of magnitude longer than for the concurrent GCs, and it is presented in log scale. Thus, Figure 5a illustrates a pause of around 4 minutes and several pauses over 10 seconds. On the other hand, for the same workload, CMS and G1 have considerably shorter pauses, going up to only 3.5 seconds (Figure 5b). Nonetheless, even though the two concurrent GCs are performing much better than ParallelOld, their pauses can still be unacceptably large on certain systems (e.g., real-time or distributed systems with critical response time between nodes).

In conclusion, we observe that the concurrency of CMS and G1 improves responsiveness and reduces GC pauses. However, this comes at the price of a lower throughput, without entirely solving the stop-the-world pauses issue. As a solution, we propose a novel GC algorithm that uses HTM to remove all synchronization pauses [4]. However, before fully implementing it we try to answer the question whether this approach would be able to maintain a comparable throughput.

2.2 HTM algorithm

Our approach assumes that the GC employs a forwarding pointer. A *forwarding pointer* normally points

to the primary copy of an object. If the object is not forwarded, the forwarding pointer is self-referential (points to itself). Otherwise, it leads the application threads to the new copy of the object.

Normally, concurrent garbage collectors allow application and GC threads to perform at the same time. In order to avoid any memory hazards when touching the same object, the application accesses (both load and store) are guarded by *access barriers*. We propose an algorithm that replaces the state-of-the-art access barriers with hardware transactions. The algorithm has the following steps:

- we encode a *busy bit* in the forwarding pointer of the object
- when an application thread needs to access an object it first starts a hardware transaction and reads the forwarding pointer of the object (to check the busy bit); thus, the address holding the pointer is automatically monitored in hardware
- before copying an object, the GC sets the busy bit (indicating that the object is in the process of being moved)
- if the application thread was accessing the same object as the GC, the write of the busy bit in the forwarding pointer will abort the started transaction; the application thread will retry when the moving is complete
- when the copying is finished, the GC clears the busy bit and updates the forwarding pointer

We implemented the algorithm on top of the ConcurrentMarkSweep GC. An important problem that we had to face was the fact that CMS injects by default barriers only for store accesses. However, our algorithm assumes access barriers for both types of accesses (i.e., load and store). Therefore, we needed to add transactional barriers for all load accesses. First, we implemented the transactional barriers and the busy bit logic in the Java interpreter.

We tested our implementation on a Intel Core i7-4770 "Haswell" CPU with fully integrated HTM support. We experimented with the same subset of DaCapo benchmarks mentioned in our performance study. We were especially interested in measuring the overhead of the newly implemented transactional barriers, in order to decide if this algorithm would bring benefits over the existing GCs. The first column in Table 1 shows the time overhead when all transactional read barriers are in place. We find that the throughput loss is unacceptable. We further observe that the write barriers have practically a negligible overhead and the values in the table are only due to the read barriers.

In consequence, we proceeded to optimize the read barriers, trying to bring down their overhead to a reasonable limit. By carefully studying the Java Memory Model (JMM) we observed that only *volatile* loads needed to return the exactly the last value written, while the other load operations could return any value previously written. We added a check for volatile loads and called the transactional barrier for this case only. The optimization greatly reduced the overall overhead, as seen in the second column of Table 1. Almost all benchmarks presented less than 4% time overhead for the optimized version of the transactional barriers (with the exception of Luindex benchmark, which had 8%).

Table 1: Transactional barriers overhead (%) in Java interpreter and JIT for the stable subset of DaCapo benchmarks

Benchmark	Interpreter		JIT
	All-read	Volatile-read	
H2	31	3.6	23.35
Tomcat	15.8	2.3	9.9
Xalan	15.06	0.5	23.11
Jython	21.29	3.5	44.98
Pmd	17.13	2.9	4.47
Luindex	26.4	8	123.07

Finally, we implemented the transactional barriers in the just-in-time compiler (JIT). The cost of the barriers is fixed regardless of the implementation, while the JITted code runs much faster than the interpreted one. Hence, the overhead of the transactional barriers becomes much more notable in the JIT compiler. The last column in Table 1 shows the relative overhead of our implementation in JIT compiler. Even though, after optimizing the read barriers, we obtained a reasonable time overhead in the interpreter, in the JIT it becomes unacceptable. All benchmarks have more than 4% overhead, going up to 123% for Luindex.

2.3 Conclusion

To sum up, we identified serious Java GC problems after an extensive study. We devised an algorithm that would replace the current synchronization mechanism, i.e., access barriers, with hardware transactions. We implemented the transactional barriers both in the Java interpreter and in the JIT compiler. We concluded that the overhead introduced by the barriers is unacceptable and impractical for a real-life GC. Even though the currently existing GCs could have important issues regarding the pause times, a garbage collector with such overhead would always perform worse than the classic GCs.

3 Work in progress and future work

Having reached a conclusion regarding the Java GCs, we moved on to a different area that might benefit from

HTM: *C++ smart pointers.*

A **smart pointer** is an abstract data type that simulates a pointer while providing additional features, such as automatic memory management or bounds checking. These features were added to classical pointers in order to reduce bugs (created by manually managing the memory), while keeping the efficiency. Smart pointers prevent most memory leaks and dangling pointers. In C++, smart pointers are implemented on top of traditional (raw) pointers, but provide additional memory management algorithms. We focus on the `std::shared_ptr` located in the `<memory>` header. A `shared_ptr` represents a container for a raw pointer, for which it maintains **reference counted** ownership. The object referenced by this pointer will be destroyed when there are no more copies of the `shared_ptr`.

First, we tried to replace the current synchronization for the reference count memory management mechanism with hardware transactions. The default synchronization uses atomic increment and atomic decrement operations, whenever a new copy of the pointer is added or removed. We modified the smart pointer implementation to use HTM and developed a first benchmark with the main goal of stressing the memory management. However, after several tests, we observed that, in this case, the transactions are too short. The overhead of starting/committing a hardware transaction is not amortized by the fully concurrent increments/decrements of the reference count.

Our current work focuses on yet another way of improving C++ smart pointers with HTM. We are looking at copies of pointers that only have the lifespan of a function. Most often, these short-lived pointers are not modified during their lifetime and thus, the memory management operations (incrementing and decrementing the reference count) are wasted. We propose to replace this pair of operations with a transaction. We implemented a new smart pointer constructor that will start the hardware transaction and consider the reference count as 0. The transaction commits when the smart pointer copy reaches the destructor (most probably at the end of the function). No operation is performed on the shared reference count. If another thread needs to modify the exact same pointer location, the transaction aborts and the transactional pointer falls back to the normal operations. Further on, we developed a simple benchmark that starts a number of threads, which repeatedly pick smart pointers out of a shared array and use them in short functions. We are in the process of checking whether in this scenario HTM improves the C++ smart pointer implementation.

As future work, we plan to further modify the smart pointer implementation in order to accommodate several short-lived pointers in the same transaction. We expect that replacing multiple pairs of atomic operations with one transaction, will amortize the transaction cost.

References

[1] M. Herlihy and J. E. B. Moss, "Transactional memory: architectural support for lock-free data structures," *Proc. 1993 International Symposium on Computer Architecture (ISCA)*, 1993, pp. 289-300.

[2] L. Gidra, G. Thomas, J. Sopena, and M. Shapiro,"Assessing the scalability of garbage collectors on many cores," *SIGOPS Operating Systems Review*, vol. 45, no. 3, pp. 15-19, 2012.

[3] M. Carpen-Amarie, P. Marlier, P. Felber, and G. Thomas, "A performance study of Java garbage collectors on multicore architectures," in *Proc. 2015 ACM International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM)*, 2015, pp. 20-29.

[4] M. Carpen-Amarie, D. Dice, P. Marlier, G. Thomas, and P. Felber, "Evaluating HTM for pauseless garbage collectors in Java," to appear in *Proc. 2015 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2015.

[5] S. M. Blackburn, R. Garner, C. Hoffmann, A. M. Khang, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J. E. B. Moss, A. Phansalkar, D. Stefanović, T. VanDrunen, D. von Dincklage, and B. Wiedermann, "The DaCapo benchmarks: Java benchmarking development and analysis," *SIGPLAN Notices*, vol. 41, no. 10, pp. 169-190, 2006.

[6] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35-40, 2010.

[7] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proc. 2010 ACM symposium on Cloud computing (SoCC)*, 2010, pp. 143-154.

The entangled storage project: Building a trustworthy long-term data storage

Verónica Estrada Galiñanes
Université de Neuchâtel
veronica.estrada@unine.ch

Abstract

The design of efficient erasure codes for storage systems is a timely topic. The storage of redundant data is a central strategy to achieve high reliability. A barrier to increase reliability is that the current methods incur in a great resources overhead. This work is an endeavour to enable efficient mechanisms to protect digital data in a trustworthy long-term data storage system. The main research question that drives this study is: How can we improve the reliability of storage systems while keeping the demand for system resources low? The thesis project is divided in four phases: (1) Understand the problem and its context; (2) Erasure code design; (3) Entangled storage; and (4) Erasure code validation. This document provides a summary of the current state of the project. First, we review our previous contributions by revisiting helical entanglement codes and give evidence of the relevance of our work. Then, we discuss the research issues that have been raised during this project and the challenges of building an storage system based on our novel codes. To conclude, we summarise the contributions to the scientific community.

Keywords: erasure codes; storage systems; reliability; data entanglement.

1 Introduction

There are multiple technical challenges in keeping data for long time. Datacenter components fail every day. Hardware and software redundancy followed by appropriate recovery plans are costly operations. Storage capacity, bandwidth, computation, energy consume, all are expensive and scarce resources in a datacenter. In addition, even though the system may follow the most up-to-date security practices, unexpected circumstances may jeopardise the data safety. Therefore, very long-term data preservation is not always guarantee or is done with caution.

During the data lifespan, it is common to apply various fault-tolerant methods for finding suitable tradeoffs that optimise the use of resources. The most trivial way of protecting data from failures is to store multiple copies of data and distribute them over different failure domains. Replication is a common approach for hot data, e.g. triplication is used in Google and Facebook datacenter. For some applications, typical user behaviour causes that data cools over time, i.e. data reads decrease in line with the age [1]. There is a large amount of literature that proposes treating warm and/or cold data with erasure coding. In contrast with replication, encoded data is dispersed across devices and reading requires contacting multiple nodes. Reed-Solomon codes are a widespread technique in storage system research. However its deployment is restricted to small subsets of data in view of the fact that if the technique is widely deployed, the network bandwidth that Reed-Solomon consumes during daily single-failure repairs will kill the normal operation of the datacenter.

The hypothesis of this research project is that the deployments of replication, Reed-Solomon, or similar techniques limit the capacity of a datacenter to survive a large amount of simultaneous failures. With the current paradigm, fault-tolerance is meant to increase proportionally to additional storage overhead.

The entire project is organised in four phases, the work accomplished in the first two phases is presented in the next section.

2 Work accomplished

2.1 Phase 1: Understand the problem and its context

The first chapters of my thesis concentrate on the multiple aspects that affect reliability in storage systems. To understand the problematic related to my research, I conducted studies and acquired some practical experience on various storage systems, some of them are Freenet, Tahoe, CEPH. Regarding erasure coding, I considered a large number of state-of-the-art approaches as well as studies that focus on design aspects that affect the reliability and availability of storage systems.

In particular, my analysis focus on data entanglement. Entanglement has been proposed more than a decade ago as an anti-censorship measure used in two systems Tangler [2] and Dagster [3]. In a nutshell, the creation of interdependencies between stored objects was proposed to generate a negative impact on the system after any tampering event. Ideally, the censoring of a document should become a very expensive operation. From the perspective of censorship deterrence, it is expected that a strong entanglement function makes that tampering one object will affect a large part of

other objects that may belong to other users. A study showed that Tangler and Dagster functions did not generate a strong graph of interdependencies among stored objects [4]. The design of a powerful algorithm for data entanglement that can be efficiently implemented is a storage system is an open problem.

Data entanglement is an interesting approach to generate redundancy in a system. Intuitively, if content is somehow connected and mixed, redundant information is dispersed in the system. Contrary to what censorship deterrent systems try to achieve, my focus is on recovering unavailable data from remaining data in the system.

The following research subquestions were addressed in the second stage of the project: (1) What mechanism can strongly entangle documents throughly with one another in an efficient manner? (2) What features should the code have to become an attractive and practical solution for cloud-based storage systems?

2.2 Phase 2: Erasure code design

Our previous work presented a novel erasure code named helical entanglement codes (HEC) to entangle files with forthcoming and previously stored data creating multiple chains of entangled data [5]. HEC embrace two ideas: interdependencies among all content in a storage system (data entanglement) and the propagation of implicit redundant information using helical chains of data and parity blocks to create a dna-inspired topology (see Fig. 6). The idea is to mix previously stored parity blocks with new data blocks and create new parities to accumulate redundant information from previous blocks. The parity block is created by XOR-ing both data and parity blocks. Each new data block always generates three parity blocks, which contribute to enlarge chains of entangled blocks (strands). The old parities are selected according to a pattern that forms a topology that resembles a helical lattice. The lattice is composed by multiple strands of data and parities. The number of helical strands is characterised with the parameter p. The new blocks continue these strands that propagate redundant information (implied redundancies). The storage overhead is 4x when data and parities are kept in the system; some optimisations are possible in certain environments, e.g. geo-replication xor coding [1].

Our recent work analyses and evaluates p-HEC in the context of cloud storage [6]. The analysis of a system in which all elements are interconnected is not a trivial task. A unique characteristic of p-HEC is that the parameter p can be tuned to achieve the desired level of fault-tolerance without adding significantly more storage overhead. Moreover, the preliminary evaluation that compares HEC with replication and Reed-Solomon shows that our solution is more fault-tolerant than traditional approaches and has many other desired properties suitable for storage systems. For instance, low locality assures that repairing single-failures does not consume large amount of resources.

3 Work in progress and future work

3.1 Phase 3: Entangled storage

An entangled storage is a system that uses HEC as the main method to store data redundantly. Building such a system has multiple challenges and requires a suitable foundation. Overall, the implementation complexity of the entanglement algorithm is low. However, there many aspects to consider when entanglement is implemented in a large distributed system. This phase is focused on two subquestions: (1) Where are blocks stored? (2) How does the network routes the requests efficiently?

The placement policy is a key factor on system's reliability. Blocks must be stored in different failure domains to avoid reducing the fault tolerance that the method offers. If a failed device contains two or more interdependent blocks, the failure will cause the loss of multiple blocks simultaneously. In comparison, the placement of a (10,4) Reed Solomon stripe requires less resources. A stripe is a collection of 10 data blocks and 4 parity blocks. Thus, it requires 14 distinct disk, nodes and ideally racks too. In HEC, placement policies require more consideration since blocks are increasingly interdependent. For instance, a 5-HEC setting requires to store 80 blocks in different devices. Block allocation is a non trivial problem and is not yet well understood in the literature.

The develop of efficient decoding algorithms to repair large amount of failures is highly dependent on block placements policies and how the requests are handled. Our project uses a distributed hash table as the storage backend. HEC requires ad-hoc algorithms to identify, store and locate the large amount of entangled blocks. Block allocation in the current DHT was designed for simple replication methods (e.g. 3-way replication) and using consistent-hashing. In contrast, all the content kept in an entangled storage is interconnected. To exploit the redundant information dissemination we need special routing and repair algorithms.

The repair traffic is estimated to be modest due to the low locality. Certain nodes can recover a missing block by requesting only one parity block from another node, and any node could recover any missing block by requesting two parity blocks.

3.2 Phase 4: Erasure code validation

In previous phases, HEC was compared with state-of-the-art codes. Simple experiments show promising results. The work plan for 2016 is to consolidate and validate all the research done during my PhD studies. As part of the validation process, I need to understand

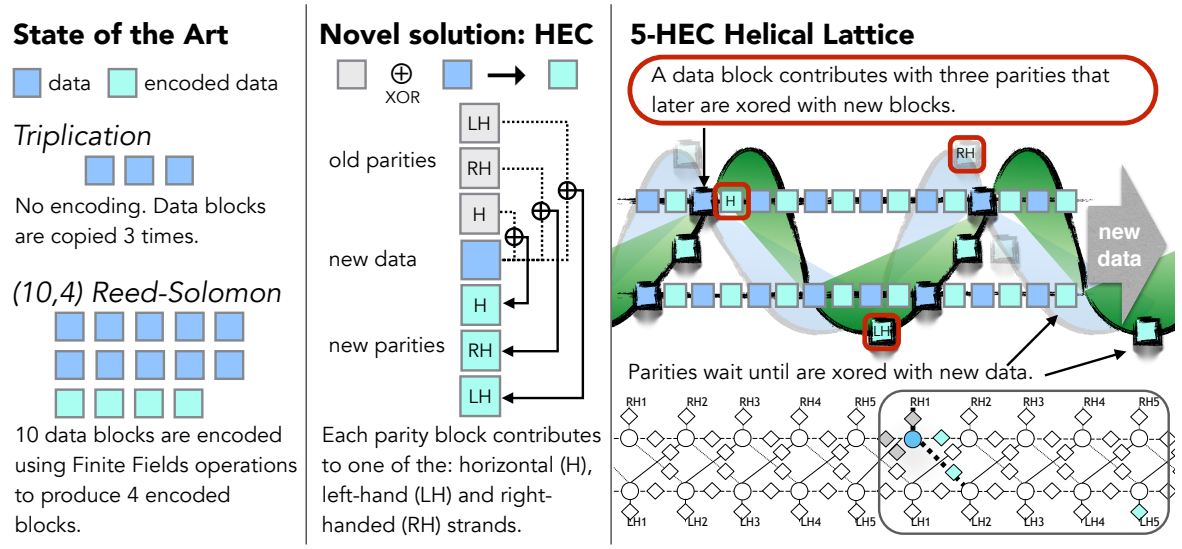


Figure 6: Traditional methods and HEC (*left*). Helical lattice in a 5-HEC (*right*).

better the benefits and the costs of implementing HEC in large-scale storage systems. To assess the feasibility of my ideas, it is necessary to obtain measurements of HEC's performance in real-world systems that will allow a realistic comparison with current redundancy methods. The development of HEC in at least one emblematic storage system will be a significant proof of the potential to become a complement solution or a valid alternative to current redundancy methods in real world systems.

4 Contributions to the scientific community

Helical entanglements are a promising solution to solve the most frequent incidents in a datacenter: single and temporary failures. Our approach provides high fault-tolerance and has many other desired properties for storage codes, e.g. low reconstruction costs, good locality. Additionally, I am working on improvements for various aspects of storage systems.

References

[1] Muralidhar, Subramanian, et al., "F4: Facebook?s warm blob storage system", *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, Usenix OSDI 2014.

[2] Waldman, Marc, and David Mazieres, "Tangler: a censorship-resistant publishing system based on document entanglements", *Proceedings of the 8th ACM conference on Computer and Communications Security*, ACM CCS 2001.

[3] Stubblefield, A. B., and Dan S. Wallach, "Dagster: censorship-resistant publishing without replication", *Tech. Rep. TR01-380*, Rice University, Dept. of Computer Science, 2001.

[4] Aspnes, James, et al, "Towards a theory of data entanglement", *9th European Symposium On Research in Computer Security*, (pp. pp. 177-192, Springer Berlin Heidelberg, ESORICS 2004.

[5] Estrada Galinanes, Veronica, and Pascal Felber, "Helical Entanglement Codes: An Efficient Approach for Designing Robust Distributed Storage Systems" *Stabilization, Safety, and Security of Distributed Systems*, pp 32-44, Springer International Publishing, SSS 2013.

[6] Estrada Galinanes, Veronica, and Pascal Felber, *(in press)* "Ensuring data durability with increasingly interdependent content", short paper, *IEEE International Conference on Cluster Computing*, IEEE CLUSTER 2015.

Service Centric Networking

Bilal Gill
University of Bern
bilal.gill@iam.unibe.ch

Abstract

Internet has evolved over the years from a communication network between different academic hosts to a content distribution network with over 1 billion connected machines. This evolution has changed the dynamics of the current internet which need a lot of patches to meet the demand of ever-growing data. To cope with these challenges, the networking research community has proposed several clean slate future internet architectures which are based on different principals such as content, services and users. One of the most promising approach is Service Centric Networking (SCN) where services are treated as first class entities. This approach is being explored by a number of research projects. In this paper, we compare and discuss features of different SCN approaches with the main focus on naming and routing.

Keywords: Content Centric Networking; Name Data Networking; Software Defined Networking.

1 Introduction

Internet in 70's was designed to provide an ability to connect hosts with resources that were located on remote hosts. This socket abstraction used IP addresses for identification which enabled different applications like FTP, telnet etc. These IP addresses were interfaces on different hosts normally on a fixed location. As a result we have a host centric abstraction, which focuses on forwarding packets among different stationary end host machines.

The current protocol stack is quite a mismatch for today's internet usage which is mostly data retrieval and service access. This stack creates a lot of hurdles and need expensive workarounds to implement the new innovations in the field of networking. For example one of the challenge, that the architecture has to provide services which are replicated in different servers for users which are increasingly mobile with a lot of interfaces. This introduces a lot of challenges in respect to multiplicity of servers and dynamism of end users. To conclude there has been no substantial changes in the layer 3 and layer 4 of the current protocol stack in the last decade to accommodate the evolution of new services and applications.

In this perspective the researchers have proposed several clean slate approaches which can resolve all the problems of the traditional internet architecture in a graceful manner. One of the approach which has found the most appraisal is Information Centric Networking which decouples the data from its location. This decoupling of the resources from the location helps to make inexpensive optimizations to the network as compared to Deep Packet Inspection in the current internet. This location independence eases the network configuration and implementation of network services.

The key principal of an ICN architecture is to name each piece of data. The named data is matched, processed and forwarded through each of the routers in the network. Names can be understood as transport layer endpoints which fetch content from the most suitable providers in the network to reduce latency. Many data centric architectures have been proposed in the past which think content based abstraction is better than host based abstraction to fulfil the demands of the current internet. Currently many projects are developing the data centric approach further, e.g CCN [6], XIA [2], NDN [7].

Most of the data centric approaches consider the static content as the main building block of their architecture. However the traditional internet not only provides content but a lot of customized services to different entities in the network like google map, load balancing, ad advertisement, transcoding and streaming video. Some services are invoked by applications, to perform different set of operations. Current Information centric architecture needs modifications to provide services to different entities in the network. These questions have given birth to a new paradigm called Service Centric Networking.

2 Related Work

There are several service centric approaches proposed previously like Serval [1], CCNxServ [4], SoCCeR [3].In all of these approaches researchers have tried to solve different problems regarding service centric networking. We will briefly describe these approaches and discuss their pros and cons.

CCNx Serv[4] was one of the first project which implemented some services on top of the CCN protocol. They gave some examples how CCN could be moulded to provide dynamic services. The name of the service is appended with the content name which is also used for forwarding. The CCNxServ enabled routers separately fetch content and service related data from the servers or replicas. The authors only target a small number of media transformational services in this paper. The implementation depends upon a NETSERV virtualization framework which provides different service modules in the network core.

SoCCeR [3] provides a mechanism to optimally select the best path towards the nearest replica or server for the content. Normally in content centric network, the interest packet is forwarded to each of the replicas in the network, which creates a lot of redundancy and network congestion. This routing scheme uses ant based colony optimization which leaves pheromones on each intermediate nodes to calculate the best path towards the server. This solution manipulates the FIB entries to set the best path along the intermediate routers.

CCNxServ and SoCCeR approaches try to eliminate the deficiencies in the CCN protocol. Some researchers are trying to incorporate information centric concepts [1] [5] in the traditional internet architecture.

For example, Function Centric Service Networking (FCSC)[5] provides flexibility to steer the flows by adding a naming layer in the SDN architecture. This naming layer provides the dynamicity, scalability and reliability to the network by reducing the interaction with the central sdn controller for setting rules in every switch of the network. Names are added at the ingress (data centre, autonomous system, ip network), where the packet automatically finds the required middle boxes with the help of the name based routing.

Serval [1] architecture provides name based sockets in comparison to host based sockets to provide modern services which require properties like multiplicity and dynamism. These name based sockets form a service access layer which is network aware and can deal with host mobility, vm migration and replica failure in the most efficient way.

3 Conclusion

In this paper, we gave a brief introduction about information centric networking and service centric networking. We discussed some related work, to understand the open problems in the field of SCN.

Most of the approaches in SCN use NDN/CCN [6] [7] protocol to provide services. CCN protocol is a bit by bit protocol which has certain semantics, difficult to obey while serving the modern complex applications. These complex applications and services require meta-information and application state at the server end. This extra information has no space in the current message (interest/data) structures of the CCN abstraction. There are some fundamental constraints which need to be addressed in a CCN jargon for Rest-ful services that drives the commercial web.

Furthermore researchers are looking how information centric concepts (caching, name based forwarding) can be applied in the current internet architecture to provide flexibility, scalability and dynamicity. These efforts include using the core principals of information centricity in data centres, software defined networks and network virtualization techniques.

References

[1] E. Nordstrom, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko,J. Rexford, and M. J. Freedman, "Serval: An end-host stack for service centric networking," " *9th USENIX conference on Networked Systems Design and Implementation.*

[2] D.Han et al, "Xia: Efficient Support for Evolvable Internetworking," " *Proc. 9th USENIX NSDI*,2012.

[3] S. Shanbhag, N. Schwan, I. Rimac, and M. Varvello, "Soccer: services over content-centric routing," " *ACM SIGCOMM workshop on Information-centric networking, ICN '11,* ,2011.

[4] S. Srinivasan, A. Singh, D. Batni, J. Lee, H. Schulzrinne, V. Hilt, and G. Kunzmann, "Ccnxserv: Dynamic service scalability in informationcentric networks," " *Communications (ICC), 2012 IEEE International Conference* ,2012.

[5] Mayutan Arumaithurai, Jiachen Chen, "Exploiting ICN for Flexible Management of Software-Defined Networks," " *ACM Conference on Information-Centric Networking (ICN 2014), Paris, France*,2014.

[6] V.Jacobson,D.K.Smetters, "Networking Named Content ," " *CoNEXT Rome*,2009.

[7] L.Zhang , et al, "Named Data Networking ," " *ACM Siggcom CCR* ,2014.

Cache Migration Strategies at the Edge of LTE Mobile Networks

André Gomes
Universität Bern | OneSource Lda.
gomes@iam.unibe.ch

Abstract

With a boom in the usage of mobile devices for traffic-heavy applications, mobile networks struggle to deliver good performance while saving resources to support more users and save on costs. Although some works already propose offloading strategies to deal with this issue, they still do not provide an effective solution for the problem as the literature shows that they can only be applied to roughly half of the requests. In this talk, we take the approach of edge caching with Information-Centric Networking and propose strategies for the pre-emptive migration of those caches at the edge of LTE mobile networks based on user mobility, content popularity and other relevant factors. With such migration strategies, the concept of content following the users interested in it becomes a reality, and content within caches is more optimized towards the requests of nearby users. Therefore, cache space is used more efficiently and cache hits increase dramatically, with clear benefits for both end users and network providers. The results of our experiments show that proposed strategies are valid and in most cases select the optimal locations and content objects for migration. Also, when compared to default cache strategies, cache hits are higher with different types of traffic, meaning that content is delivered faster while saving a large quantity of bandwidth at the core of the network.

Keywords: Information-Centric Networking, Content Migration, Caching, LTE.

1 Introduction

Mobile network evolution in the last few years has been quite intense, with major increase of throughput performance and resources usage efficiency. Such evolution is mostly driven by tremendous demand of bandwidth [1], on the one hand because smartphones and other mobile devices play a major role as content demanders, and on the other hand because traffic-heavy applications are part of the daily life of millions of people. However, satisfying the content requirements of the current number of users with such dynamic networks is still an open challenge, which is currently being addressed by a number of emerging concepts and technologies.

As far as the network is concerned, new 5G concepts such as Network Function Virtualization (NFV) [2] are emerging, allowing networks to adapt more dynamically to different conditions and requirements. One of these efforts is Cloud Radio Access Network (C-RAN) [3]. It brings the possibility to virtualize the entire 3GPP LTE radio infrastructure, except for the antennas. Virtualized infrastructures extend the cloud computing concept to the Radio Access Network (RAN), and explore the modularity of the components together with the usage of general-purpose hardware infrastructure to run evolved Node Bs (eNBs). Such fact transforms C-RAN into an enabler for deployment of value-added services closer to the edge of the network, i.e. in very close proximity to mobile users.

At the same time, Future Internet (FI) concepts such as Information-Centric Networking (ICN) or Content-Centric Networking (CCN) [3], which propose a change in the current paradigm of requesting content, are becoming increasingly important. Such fact is explained by the advantages brought together. Currently, when a user wants to request a content object, it has to query one specific server to request a given resource. With CCN, the user has no knowledge about servers and just requests the network for content. It first sends an Interest message that usually specifies the content provider and the name of the content object. This message is then routed by CCN routers, which have forwarding tables based on content naming. These routers will find the content object (if available) and return it directly to the user, without any further interaction. This approach has multiple advantages over traditional Internet Protocol (IP) networks, especially in terms of performance. This is truly relevant in current mobile networks where resources are scarce and traffic loads have exploded. Indeed, with CCN content retrieval is much quicker due to faster lookups and because of the inherent caching, which is native.

With such concepts in mind, proposals appear to take advantage of their combination. Gomes et al. [5] evaluate the feasibility of deploying ICN/CCN together with LTE mobile networks, leveraging the C-RAN concept and its role as an enabler for the deployment of additional services within these networks. In that work, authors conclude that there are clear benefits of deploying CCN routers co-located with LTE eNBs, such as bandwidth savings at the core network and lower latency to retrieve content derived from the proximity to end users.

With CCN-based caching at the edge of 3GPP LTE networks at its inherent benefits, one may assume that major performance improvements and reduction

of traffic at the core of the network are already achieved by default. However, efficient caching strategies are needed in order to populate those caches and maintain content where it will yield the most benefit. As users are increasingly mobile and tend to move between different locations quite often, it is safe to assume that what is cached at a location is not necessarily what is going to be requested by the users that will be there in the next few hours or days. That fact leads to the question: how does user mobility affect caching strategies? Such question does not have a simple answer, as some assumptions have to be made and challenges need to be considered to reach a preliminary conclusion. First, it is important that user mobility is predicted to perform preemptive actions. Then, if a set of users is predicted to be at a location, decisions need to be made. The first decision is whether content from caches at the origin of the users should be migrated to other caches at the possible destinations. Once that is established, two other questions arise: where should content be migrated to (mobility prediction usually outputs a list of possible destinations with different probabilities) and which subset of the content should be migrated.

In our proposal, described in the next section, we attempt to answer the previously highlighted questions by developing content migration strategies that handle the required decisions and deliver the greatest possible tradeoff between benefit and cost.

2 Follow-Me Cloud Strategies

In-line with the idea of content/services following users - Follow-Me Cloud (FMC) - our proposal can be summarized into making decisions and perform preemptive migrations of mobile network's edge-cached content based on user mobility, i.e. migrations ahead of future user requests at a new location. The following key objectives are assumed: mobility prediction must be used to take actions before users move from one location to another, content may only be migrated if it will yield benefit at the destinations, multiple destinations may be considered at the same time to improve accuracy and migration cost should be minimized as much as possible. We also assume that content migrations should happen between repositories, as caches have limited size due to resources constraints, are not persistent (stored in volatile memory) and typically implement a Least Recently Used (LRU) policy that can delete recently added content in a matter of seconds if the load of received Interests is high.

As far as decisions are concerned, two types of decisions must be carefully analyzed and made in the proposed system. The first decision is **where** to migrate content when users are moving. Although mobility prediction will output a list of candidate destinations and respective probabilities, such set of destinations may not be complete and still needs to be reduced and ranked according to other important factors.

Multiple-Criteria Decision-Making (MCDM) [6] is an approach to make decisions in the presence of multiple, usually conflicting, criteria. As it is not tied to a specific problem, it can be applied to a very diverse range of scenarios and problems, from business decisions to complex science problems. At the same time, it supports multiple weighted criteria and typically returns a finite number of solutions when dealing with a selection/assessment problem. Therefore, it fits the decision to be performed in terms of ranking/selection the destinations for content migration.

The first step of the decision is to define criteria, and in this case the following were defined:

- Mobility prediction information containing accuracy/probability.
- Percentage of non-intersecting content between origin and alternative destination.
- Percentage of free storage space at alternative destination.
- Relative size of mobile group, defined as the ratio between number of users moving and users present at alternative destination.
- Cost of migration, defined as network delay between origin and alternative destination.

As each of the criteria may have a different significance for the decision, each of them should also have a weight value to be considered. We may define different weight sets depending on the scenario and the relevance given to each of the criteria. With the scores of each alternative for each criterion and the respective weights, the score of alternative i is given by:

$$S_i = \sum_{\substack{j=1 \\ \forall i \in [1,N]}}^{M} w_j r_{ij} \tag{1}$$

where:

S_i is the score of the i^{th} alternative;
r_{ij} is the normalized rating of the i^{th} alternative for the j^{th} criterion, which is calculated as $r_{ij} = x_{ij}/(\max_i x_{ij})$ for benefits and $r_{ij} = \frac{1}{x_{ij}}/(\max_i \frac{1}{x_{ij}})$ for costs;
x_{ij} is an element of the decision matrix, which represents the original value of the j^{th} criterion of the i^{th} alternative;
w_j is the weight of the j^{th} criterion;
M is the number of criteria;
N is the number of alternatives.

With the list of destination alternatives ranked and sorted in descending order by their score, hereafter just called "ranking", the decision about where to migrate content may be made based on the defined policies. For instance, the first 3 alternatives (destinations) of

the ranking can be selected and content will be migrated to all of them. Or, depending on the problem and assuming that the score is normalized, alternatives with scores above 0.75 are to be selected.

After the decision about the destinations has been taken, the remaining decision of **what** content should be migrated still needs to be taken. It takes the local popularity of content as key criterion and considers the following steps. First, if the content object being considered is available nearby (1 hop distance) or already at the destination, it will not be migrated. Second, if it is not available, and if free space is available at the destination, content objects are just migrated until the cache is filled. Third, if no free space is available, both popular content at the origin and destination should be considered together and ranked to fill the destination cache with the content that will deliver the greatest benefit for all the users (existing and new ones). The following equation was considered to calculate the score of each content object k:

$$S_k = \begin{cases} p_k * \dfrac{n_{mgt}}{n_{mgt} + n_{dst}}, \text{if the } k^{th} \text{content object is} \\ \text{considered for migration.} \\ \\ p_k * \dfrac{n_{dst}}{n_{mgt} + n_{dst}}, \text{otherwise.} \end{cases} \tag{2}$$

where:

S_k is the score of the k^{th} content object;

p_k is the local popularity of the k^{th} content object;

n_{mgt} is the number of users migrating from origin to the selected destination;

n_{dst} is the number of users at the selected destination for content.

With the content objects ranked and sorted in descending order by their score, content is selected to fill the cache until its size threshold. When content is not available locally, the CCN router at the destination will fetch it from the origin of from a closer router (if available).

3 Evaluation and Results

In order to evaluate the proposal described in the previous section, a realistic mobility trace was selected [7]. This trace includes data from one hundred human subjects over the course of nine months, and it was collected by MIT students using Nokia 6600 smart phones in the academic year of 2004/2005. Although the information collected includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status, for this evaluation only the information on mobility was considered: Object Identifier (OID), endtime, starttime, person_OID, celltower_OID. Therefore, it is possible to know at every given time to which cell a given person is connected. Such trace can thus be used to assess how the system behaves in realistic conditions, as if mobility was generated in any other way, it would probably create biased results and render the conclusions invalid for real-world scenarios.

Concerning mobility prediction, as it is not the focus of this work, 1-hour delta time predictions (one hour in the future) were generated at every 1 hour of simulation time according to the results obtained by mobility prediction works. As with 50% of user movement randomness the accuracy achieved is also around 50%, the generated predictions had an accuracy following a normal distribution $N(50, 10)$.

To run experiments, mobility data was fed into the ns3 simulator and links were created between CCN user nodes (devices) and CCN router nodes, with each cell containing its own CCN router node. Each CCN router node consists of the main daemon (CCNd) and a repository (CCNr). As the selected CCN implementation, the widely adopted CCNx, is not available directly within ns3, ns3-dce was used to run the CCN nodes with the full CCNx protocol implementation. At the same time, due to the large set of data (32GB), computation was very intensive to run in our local test bed, so a Linux cluster [8] was adopted as the platform to run all the defined experiments and every experiment was run 30 times with a different, randomly selected week of trace mobility data.

Assuming that different weights for the criteria may return different results, four different weight sets were considered for evaluation. A first weight set (weight set #1) gives priority to mobility prediction information and group size, while the second (weight set #2) shifts priority to the percentage of non-intersecting content between origin and alternative destination together with free space at destination. At the same time, 2 extra weight set exist based on the initial ones (weight set #1 with cost and weight set #2 with cost). They are similar to their basis, but also include cost as a factor. Here cost is defined as the distance (hop count) between origin and possible destination of content.

The evaluated metric is the position of an optimal solution (highest profit destination) in the score-sorted ranking of destination alternatives derived from the output of the decision making algorithm. The optimal solution is the location where the group of users was (within a time interval of 2 hours since the prediction) on which the requested volume of content recently migrated was the highest. If it is in the first positions (1, 2, 3, etc.) of the aforementioned ranking, it means that the decisions were good and will yield benefits for the users.

To assess that metric, a comparison is made between the different weight sets, and a Cumulative Distribution Function (CDF) is generated for each ranking position. With the CDF, it is possible to obtain the cumulative percentage of times when the optimal solution was within the n first positions of the ranking. For ex-

ample, one may assume that x percent of the times the optimal solution was at the first three positions of the ranking of destinations.

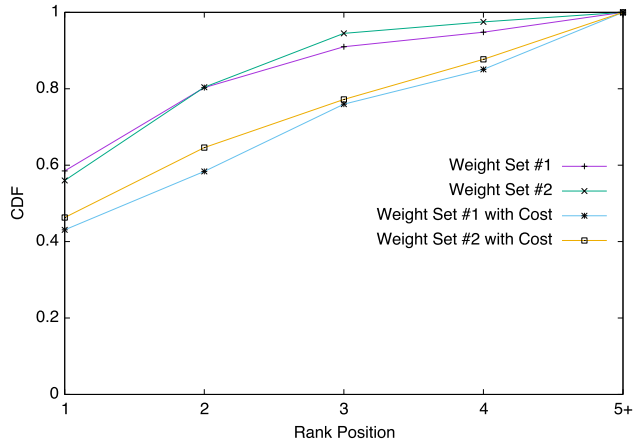


Figure 7: Optimal Solution in Ranking

In Fig. 7, results show that weight set #1 has a higher percentage of optimal solutions at the first position of the ranking, meaning that selections were perfect in almost 60% of the cases. However, the results of weight set #2 converge quicker to 100% of the cases, surpassing weight set #1 after (and including) rank position number 2. Overall, one may assume that weight set #2 selects better options than any other weight set, especially considering that the optimal solution was within the first three positions of the ranking in more than 90% of the cases. This can be easily explained by the accuracy of mobility prediction, which can vary immensely and does not account for the time users spend at the predicted locations. At the same time, giving priority to destinations where most of cached content is not the same as in the origin has a big inherent potential to be explored from the beginning.

When looking at the weight sets but considering cost, the trend is slightly different. Weight set #2 with cost outperforms weight set #1 with cost from the beginning, with the optimal solution being in the first position of the ranking almost 50% of the cases and in more than 80% of the cases the optimal solution being in the first 4 positions of the ranking. These results are according to what was expected, as cost limits the performance but considering it still delivers a good tradeoff for both end users and network operators.

4 Conclusions and Future Work

In this work, concepts and strategies for the migration of content within mobile networks were introduced, enabling multiple benefits both from the user and network perspective. As the users move to different locations, they still want to access content on which they are interested with a low latency and without delays or breaks, especially if dealing with multimedia content. From the network perspective, this can only be granted if caches exist at the edge of mobile networks and content kept in those caches (with limited resources) is the right content, i.e. popular content that local users are very interested on.

Our proposal to address these requirements was evaluated in terms of performance, considering multiple weight values and different scenarios. Although the results can be considered quite good, improvements can still be made. For instance, more advanced strategies can be used to decide which subset of content should be migrated. We envision that popularity may not be the only factor to decide which content to migrate due to is general nature, but also other factors that relate user to content should be considered in future work.

References

[1] Cisco Systems Inc. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20142019. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf`, February 2015.

[2] Demestichas, P.; Georgakopoulos, A.; Karvounas, D.; Tsagkaris, K.; Stavroulaki, V.; Jianmin Lu; Chunshan Xiong; Jing Yao, "5G on the Horizon: Key Challenges for the Radio-Access Network," Vehicular Technology Magazine, IEEE, vol.8, no.3, pp.47,53, September 2013.

[3] Bernd Haberland, Fariborz Derakhshan, Heidrun Grob-Lipski, Ralf Klotsche, Werner Rehm, Peter Schefczik, and Michael Soellner. Radio Base Stations in the Cloud. Bell Labs Technical Journal, 18(1):129–152, 2013.

[4] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In Proc. of the 5th international conference on Emerging networking experiments and technologies (CoNEXT '09). New York, NY, USA, 1-12, 2009.

[5] Andre Gomes and Torsten Braun. Feasibility of Information-Centric Networking Integration into LTE Mobile Networks. In Proc. of the 30th Annual ACM Symposium on Applied Computing, SAC '15, pages 628–634, April 2015.

[6] Dong-Ling Xu. An introduction and survey of the evidential reasoning approach for multiple criteria decision analysis. Annals of Operations Research, 195(1):163–187, 2012.

[7] Nathan Eagle and Alex (Sandy) Pentland. Reality mining: Sensing complex social systems. Personal Ubiquitous Comput., 10(4):255–268, March 2006.

[8] University of Bern. University of Bern Linux Cluster (UBELIX). http://www.ubelix.unibe.ch, May 2015.

Exploiting Caching to Improve the Performance of Geographically Distributed Stores

Raluca Halalai
Université de Neuchâtel
raluca.halalai@unine.ch

Abstract

Many contemporary distributed applications rely on large-scale data stores that are expected to provide high availability at low access latency and low storage overhead. A storage system ensures high availability by employing redundancy schemes, which influence the access latency and the storage overhead. In this paper, we present an overview of the state-of-the-art approaches to building highly available data stores, and discuss their effects on the access latency and storage cost. We present our efforts towards developing a storage infrastructure that aims to fulfill the previously described requirements by combining two well-known redundancy schemes – replication and erasure coding, and discuss the challenges it poses.

Keywords: storage systems, high availability, low storage cost, low access latency, cache coherence

1 Introduction

Modern distributed applications have millions of users worldwide and need to deal with petabytes of data. Thus, such large applications rely on data stores that aim to ensure:

- *high availability:* users can access their data despite faults, ranging from intermittent problems to disasters that take down entire data centers;
- *low access latency:* users can access their data fast;
- *low storage cost:* the system does not store too many copies of the data in order to achieve the first two goals.

LAN data stores, which are deployed at one data center from one geographical location, cannot provide the guarantees that modern applications strive for. Instead, these applications rely on data stores that span many data centers at geographically diverse locations; we refer to these as *geo-distributed stores*.

Geo-distributed stores rely on redundancy to ensure high availability in the presence of failures, which may range from a node crash to the unavailability of an entire data center. The redundancy schemes that are typically used in storage systems are replication and erasure coding. [1]

Replication schemes store copies of data both within and across data centers; this provides low access latency, as clients are able to access the nearest replicas, but it is costly in terms of storage space and potentially bandwidth, as replicas need to be kept in sync.

Erasure coding can provide similar availability guarantees at lower storage cost, but encoding, decoding, disseminating and, respectively, gathering data is expensive. Thus, erasure coding results in significantly higher access latency and bandwidth overhead.

The choice of redundancy scheme determines the access latency and the storage cost. Table 2 shows how replication and erasure coding impact access latency and the storage overhead. The goal of this work is to leverage replication and erasure coding, in order to get the best of both worlds: (i) lower storage overhead by storing data encoded, (ii) lower bandwidth consumption by storing data also in decoded form, and (iii) lower access latency by replicating data partially, such that it is close to the clients that use it.

Property	Replication	Erasure coding
High availability	Yes	Yes
Low access latency	Yes	No
Low storage cost	No	Yes

Table 2: Comparison of replication and erasure coding.

2 Work accomplished

This section provides an overview of the work we have accomplished so far. First, we present the design of a system that combines the redundancy schemes discussed in Section 1. Second, we discuss the main challenge that such a design poses, and present some potential solutions to overcome them.

2.1 Conceptual design

The problem of combining fast, but expensive storage with cheaper, but slower storage in an efficient manner has already been addressed in computer architecture, where memory is organized in a hierarchy. Figure 8 shows a conceptual representation of the memory hierarchy within a computer with multiple processing cores. *Static RAM* (SRAM) is fast, but expensive – like replication, while *Dynamic RAM* (DRAM) is slower, but cheaper – like erasure coding. In the memory hierarchy of a computer, SRAM is used to implement *caches*. A cache is a storage component that provides fast access to data that has been recently used.

Since workloads exhibit temporal and spatial locality, caching the most recently accessed data provides the illusion of most data being stored in fast memory.

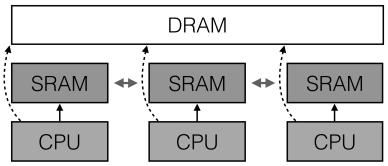


Figure 8: The memory hierarchy within a computer.

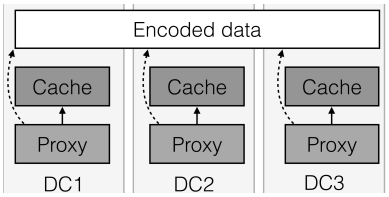


Figure 9: The hierarchical design applied to a geo-distributed data store.

Previous studies [2] have shown that also workloads specific to geo-distributed systems exhibit temporal locality of reference. Hence, the conceptual design of a memory hierarchy presented in Figure 8 can be extended to a data store tailored to work across geographically-diverse data centers, as shown by Figure 9. In the context of a geo-distributed data store, the slower, but cheaper storage component is a key-value store that writes encoded data to disk, and is deployed across data centers. We envision the cache component as a collection of independent nodes that store decoded data in RAM. This design exploits locality of access by having a caching component deployed in each data center; this implies that data may be partially replicated across data centers.

2.2 Challenge

A hierarchical design that uses multiple cache components poses the following challenge: *how to ensure coherence among* (i) *different caches*, (ii) *caches and the slower storage component?*

Cache coherence in a computer. There are three ways to deal with the cache coherence problem in a memory hierarchy: 1) avoid the problem by not caching data that is shared among CPUs or by not caching mutable data; 2) if there is a shared bus, use snoopy caches that monitor the bus for changes to the data they store, and invalidate or update it; 3) if this assumption does not stand, emulate a shared bus using an atomic broadcast protocol, and use some directory-based protocol that allows a cache to communicate only with other concerned parties.

Cache coherence in a geo-distributed store. A geo-distributed store can deal with the cache coherence challenge in the following ways: 1) avoid the problem by confining each data item to a primary data center; 2) replicate the cached data at all data centers, and use a consensus protocol (e.g., Paxos [3], Raft [4]) to ensure total order among update operations that impact cached data; 3) partially replicate data across data centers and adapt a cache coherence protocol to keep replicas in sync.

The first two approaches for achieving consensus in a geo-distributed environment have known pitfalls: assigning each data item to primary data center does not exploit locality of access, and fault-tolerant consensus in a geo-distributed setting is notoriously expensive [5]. Intuitively, the third approach is also costly in terms of performance, but, to the extent of our knowledge, there is no recent study on this topic. In the context of a geo-distributed data store, we believe that it might be possible to combine replication and erasure coding in a way that would minimize the performance cost of consensus.

Keeping the cached data in sync with the data stored by the slower storage component is an additional challenge. In computer architecture, a cache can implement three mechanisms to solve this issue: 1) *write-through:* write operations update both the cached data and the data stored by the slower memory; this approach is good for workloads that exhibit temporal locality of access; 2) *write-around:* write operations bypass the cache, and are executed only by the slower memory; this approach reduces the contention at caches, but it does not leverage temporal locality of access; 3) *write-back:* write operations are only executed by the cache, bypassing slower memory; this approach improves latency for write-intensive workloads, but might result in decreased availability. Each of these mechanisms can be applied to a geo-distributed store in a straightforward manner, and ,thus, we will not discuss them further.

3 Work in progress and future work

In this section, we discuss our efforts towards designing a system based on the conceptual design presented in Figure 9, and implementing a first prototype that allows us to easily experiment with redundancy schemes and coherence protocols.

In Figure 10, we present the design of a data store that combines the replication and erasure coding redundancy schemes. This data store is intended to be deployed across multiple data centers, located in different geographical regions. The main components of the system are:

- a persistent key-value store (e.g., Redis [6] or Voldemort [7]) that spans multiple data centers, and stores blocks of encoded data on persistent media;

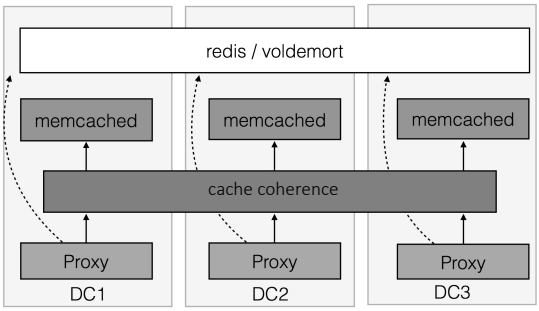


Figure 10: Design of a geo-distributed storage system that uses caching.

- a pool of Memcached [8] nodes per data center, storing in RAM the most recently accessed data, in decoded form;
- a pool of proxy nodes that listen for client requests, and are able to: (i) contact the local cache and the persistent store, (ii) initiate the cache coherence mechanism, and (iii) make the use of erasure coding transparent to the client;
- a cache coordination mechanism to keep in sync data stored at multiple caches.

We are currently working on implementing a flexible prototype based on the previously described design. As a next step, we intend to study the challenges of achieving consensus in a geo-distributed environment, experiment with different consensus algorithms, and different approaches of combining the two redundancy schemes – replication and erasure coding.

4 Conclusion

This on-going work addresses the problem of building a geo-distributed store that is highly available, and provides low access latency at relatively low storage cost. To this end, we borrow an idea from computer architecture: organize the data store in a hierarchy with two layers – one fast, but more expensive in terms of storage cost, and one slower, but cheaper in terms of storage overhead. We are working on finding efficient ways of dealing with the challenge posed by such a hierarchical design: how to maintain data coherent?

References

[1] R. Rodrigues and B. Liskov, "High Availability in DHTs: Erasure coding vs. Replication", in *Peer-to-Peer Systems IV 4th International Workshop (IPTPS)*, 2005.

[2] Y. Ma, T. Nandagopal, K. Puttaswamy, S. Banerjee,"An Ensemble of Replication and Erasure Codes for Cloud File Systems", in *IEEE INFOCOM*, 2013.

[3] L. Lamport, "Paxos Made Simple", in *SIGACT News*, Vol. 32, No. 4, 2001.

[4] D. Ongaro, J. Ousterhout, "In Search of an Understandable Consensus Algorithm", in *Proceedings of the 2014 USENIX Annual Technical Conference*, 2014.

[5] B. W. Lampson, "How to Build a Highly Available System Using Consensus", in *Proceedings of the 10th International Workshop on Distributed Algorithms (WDAG)*, 1996.

[6] "Redis", `http://redis.io/`, accessed on July, 7 2015.

[7] "Project Voldemort", `http://www.project-voldemort.com/voldemort/`, accessed on July, 7 2015.

[8] "Memcached", `http://memcached.org/`, accessed on July, 7 2015.

Seamless utilisation of GPU resources for runtime and energy reduction in the Actor Model

Yaroslav Hayduk
Université de Neuchâtel
yaroslav.hayduk@unine.ch

Abstract

The common way of defining concurrent programs is to use threads and locks. There are problems associated with the usage of locks, which are error prone and might not scale, if not properly used. The Actor Model provides inherent parallelism and safeguards against common concurrency hazards, such as race conditions and deadlocks. It has been successfully used for building large-scale distributed applications, capable of simultaneously utilizing multi-core processors and grid and cloud resources. The model consists of entities, called actors, communicating with asynchronous message passing. My research is focused at improving the model's (1) performance, (2) energy efficiency, as well as its (3) programmability. The model's scalability is hindered as the Actor Model enforces actors to process messages sequentially to provide safety guarantees. To increase message throughput, we processed messages concurrently by using transactional memory. Later, we demonstrated that message throughput could be further increased by dynamically selecting the optimal number of messages according to the contention level. Given that $E = P * T$, for a wide range of data parallel applications, GPUs can radically reduce the execution time (T) while marginally increasing the power (P), as compared to executing on the CPU. As a result, heterogeneous programs yield lower energy consumption (E). Currently there is no simple way of specifying heterogeneous code, as the programmer needs to use different programming languages and be aware of the differences in memory models. Therefore my current research is focused at enabling the seamless use of GPU resources in the context of the Actor Model. **Keywords:** concurrency; actors; transactional memory; heterogeneity.

1 Introduction

The Actor Model, initially proposed by Hewitt [3], is a successful message passing approach that has been integrated into popular local and distributed frameworks [4]. It addition to being able to avoid common concurrency hazards associated with using locks (e.g., data races, deadlocks), the Actor Model provides means to define safe, reliable, concurrent and maintainable programs. An actor is an independent, asynchronous object with an encapsulated state that can only be modified locally based on the exchange of messages. Received messages are processed sequentially. The Actor Model introduces desirable properties such as encapsulation, fair scheduling, location transparency, and data consistency to the programmer.

The Actor Model is inherently parallel, meaning that once actors, the main entities in the Actor Model, are defined, the user does not need to put additional effort to parallelize their execution. In my research we have identified a number of issues in the Actor Model, which limit its performance, energy efficiency, as well as its programmability. The issues are as follows: (1) sequential message processing, (2) blocking coordinated message processing, (3) concurrent message processing in cases of high contention, (4) underutilization of resources, (5) no support for definition of heterogeneous actors, and (6) no support for scheduling of heterogeneous actors.

2 Work Accomplished

In the following sections we will describe the problems for which we have already devised solutions for in more detail.

2.1 Sequential message processing [1]

Despite its inherent concurrency, the Actor Model requires sequential message processing. While this requirement is a deliberate choice to avoid synchronization hazards, it unnecessarily limits the performance (i.e., throughput) of individual actors, in particular when they execute on multi-core computers. We claim that the delays associated with sequential processing *impede the model's performance*; to *improve the model's performance* we process messages concurrently. By encapsulating the handling of messages inside Transactional Memory(TM) transactions, we preserve the atomicity and isolation guarantees offered by the Actor Model. Thanks to the rollback and restart capability of transactions, several messages can be processed concurrently, even if they access the same state.

2.2 Blocking coordinated message processing [1]

Any coordination between actors requires a distributed transaction, which we call *coordinated transaction*. When executing a coordinated transaction, all the actors that have finished their part of the work need to block and wait until other actors, which are members of

the same coordinated transaction, have completed processing. When actors block and they cannot perform any useful work, hence *negatively affecting the performance* of message processing. As such, we claim that the delays associated with blocking are unnecessarily long and can be significantly reduced by thoughtful changes to the message processing of actors.

To improve the model's performance, we exploit the speculative properties of transactions. We remove blocking and speculatively process next messages from the actor's mailbox. If the speculatively processed message does not interfere with the coordinated transaction, it can be safely processed. We block the speculative message processing and further message processings for a given actor only when we detect a conflict between the speculative message processing and the coordinated transaction (i.e., the speculative message processing tries to modify variables accessed as part of the coordinated transaction) .

2.3 Concurrent message processing in cases of high contention [2]

In cases of high contention, the performance of concurrent message processing can drop close to or even below the performance of sequential processing. *To improve the performance* of such high contention workloads, we propose to determine the optimal number of threads that execute transactional operations as the performance of an application is dependent on the level of concurrency. To avoid high rollback counts in high contention phases fewer threads are desired, in contrast to low contention phases where the number of threads can be increased.

At the beginning of an application we start from a random number of threads from the thread pool to process transactional messages (STM messages) and then, driven by a predefined threshold α, the level of concurrency is adapted. The dispatcher assigns the rest of the threads to process read-only messages. α is dependent on the knowledge of the current number of commits and rollbacks of transactionally processed messages and their ratio. Instead of focusing on the throughput such as Didona et al. [7], we are able to support transactions of any granularity by steering α with the commit-to-rollback ratio. If the current commit-to-rollback ratio is lower than α, we divide the number of threads processing transactional messages by two; if it is higher, we multiply them by two. To find the right α value for the current workload, we consider a short profiling phase monitoring the relation of commits and rollbacks.

2.4 Underutilization of resources [2]

To avoid high rollback counts, the aforementioned approach reduces the number of actively used threads, leaving some threads idle, which, in turn, leads to the underutilization of the system. Further, this also *affects the model's power consumption* in that when the cores are left idle, they still consume a non-trivial amount of energy. In order to use idle system resources to *increase the model's performance as well as its energy efficiency*, we propose to extract messages for which we can relax the atomicity and isolation guarantees and process them as non-transactional messages. We call these messages read only messages.

In our experiments we used Scala STM as the default STM implementation. Scala STM uses *Ref* objects to manage and isolate the transactional state. The Ref object does not permit accessing its internal state in a non-transactional context. Our proposal is to break the isolation guarantees of the Ref object in specific cases, i.e., if we limit the reads to specific isolated values, we can omit using the transactional read and grant direct access. By using direct access for a number of scenarios in the context of the Actor Model, we can process a substantial amount of read-only messages while not conflicting with messages processed in regular transactions.

3 Work in Progress and Future Work

In the following sections we will outline the issues that we are in the process of solving as well as discuss open questions.

3.1 No support for definition of heterogeneous actors

The latest trend in creating modern systems is heterogeneity. For example, besides having a CPU, modern laptops are typically equipped with both, integrated (on the same die as the CPU) GPU and stand-alone GPU. In its current incarnation, however, the Actor Model supports seamless execution only on a CPU.

Given that E=P*T, for a wide range of data parallel applications, GPUs can radically reduce the running time (T) while marginally increasing the power (P), as compared to executing on the CPU. This power-runtime tradeoff typically yields lower energy consumption. As such, it would be beneficial to enable Actor Model users to harness the GPU resources in order to *enhance the energy efficiency of the model.* Currently, to exploit the GPU (assuming that Akka [11] is used), Actor Model users are required to consider solutions based on JNI [8] or RabbitMQ [9]. The main issue with these solutions is that they are not directly supported by the Actor Model, in the sense that the user has to deploy them from scratch, with can be cumbersome and time consuming. As a results *programmability of the model suffers. To enhance programmability*, we propose to add support for defining the message processing code in the Actor Model using a Domain Specific Languages (DSLs) as provided by the Delite Framework [10]. The Delite framework allows to exploit het-

erogeneous resources; it works as follows: (1) users develop applications in the provided DSLs; (b) once the program has been defined, Delite supports the generation of efficient Scala, C++, CUDA and OpenCL code. Delite, however, does not allow its efficiently generated code to run outside its runtime. As such, the main challenge is to support the definition and execution of Delite programs inside Actors without requiring the users to use the Delite runtime.

3.2 No support for scheduling of heterogeneous actors

Assuming that users have the capacity to define the code for processing the same message on the CPU and the GPU, the main challenge is to select the most appropriate version for the to be processed message. Choosing the most appropriate actor type is important, as this discussion directly *affects not only performance, but also energy efficiency.*

Given a pool of (similar) tasks and a number of actors running on the GPU and the CPU, we foresee using a design that is based on work stealing. Once an actor completes processing all the messages in its mailbox, he will attempt to steal unprocessed messages from other actor's mailboxes.

Since many CPU threads can be assigned to submit tasks to the GPU, we can start not one but many GPU actors, which can concurrently submit work on the GPU. As such, in a setup which supports executing multiple CPU and GPU actors, it is not clear how to select the optimal number of actors of each type such that the processing time of the available messages is minimized. In addition, as the GPU has its own memory, the challenge is to automatically migrate memory from the CPU to the GPU (or visa versa), if our scheduling decision dictates to use more GPU actors. Lastly, in the context of executing iterative application, when running on the GPU, the challenge is provide support for specifying which data needs to be kept on the GPU across multiple iterations.

4 Conclusion

In its current incarnation the Actor Model, while being an efficient solution for defining and executing concurrent applications, contains a number of issues, which affect its (1) performance, (2) energy efficiency, as well as its (3) programmability. The issues (and descriptions of what they affect) are as follows: (a) sequential message processing (performance); (b) blocking coordinated message processing (performance); (c) concurrent message processing in cases of high contention (performance); (d) underutilization of resources(performance and energy efficiency); (e) no support for definition of heterogeneous actors (programmability) and (f) no support for scheduling of heterogeneous actors(performance and energy efficiency). We have solved and published the solutions for the first four issues [1, 2].

References

[1] Y. Hayduk, A. Sobe, D. Harmanci, P. Marlier and P. Felber "Speculative Concurrent Processing with Transactional Memory in the Actor Model," in *Proc. 17th International Conference on Principles of Distributed Systems (OPODIS)*, 2013, pp. 160–175.

[2] Y. Hayduk, A. Sobe and P. Felber "Dynamic Message Processing and Transactional Memory in the Actor Model," in *Proc. The 10th International Federated Conference on Distributed Computing Techniques (DAIS)*, 2015, pp. 94–107.

[3] C. Hewitt, P. Bishop, R. Steiger, "A universal modular actor formalism for Artificial Intelligence," in *Proc. of the 3rd international joint conference on Artificial Intelligence (IJCAI)*, 1973, pp. 235–245.

[4] R. K. Karmani, A. Shali, G. Agha, "Actor frameworks for the JVM platform: A comparative analysis," in *Proc. of the 7th International Conference on Principles and Practice of Programming in Java (PPPJ)*, 2009, pp. 11—20.

[5] P. Haller, "On the integration of the Actor Model in mainstream technologies: The Scala perspective," in *Proc. of the 2nd edition on Programming systems, languages and applications based on actors, agents, and decentralized control abstractions (AGERE!)*, 2012, pp. 1-–6.

[6] B. J. Smith, "Shared memory, vectors, message passing, and scalability," *Parallel Computing in Science and Engineering*, vol. 25, no. 3, 1988, pp. 27-–34.

[7] D. Didona, P. Felber, D. Harmanci, P. Romano, and J. Schenker. "Identifying the optimal level of parallelism in transactional memory systems." in *Proc. of the 1st International Conference on Networked Systems (NETYS)*, 2013, pp. 233—247.

[8] Java Native Interface, available at `http://docs.oracle.com/javase/7/docs/technotes/guides/jni`

[9] RabbitMQ — an open source message broker software, available at `http://www.rabbitmq.com`

[10] Delite — a compiler framework and runtime for parallel embedded domain-specific languages (DSLs), available at `http://stanford-ppl.github.io/Delite`

[11] Akka — a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM, available at `http://akka.io`

Energy Efficiency in Heterogeneous Data Centers

Mascha Kurpicz
Université de Neuchâtel
mascha.kurpicz@unine.ch

Abstract

Cloud computing as a whole consumes more energy than entire countries like Germany or India. Due to the environmental concerns of different energy resources and the massive power consumption of large data centers, energy efficiency becomes more and more important.

In my work, I want to achieve reduced energy consumption in data centers by providing tools under the consideration of different aspects: (1) power estimation, (2) profiling of heterogeneous environments, and (3) analysis of co-located workloads.

The basis for energy-efficient data centers is measurement and estimation of power consumption. To address this, we developed BitWatts, a middleware toolkit for process-level power estimation on physical machines and virtual environments.

In data centers we usually comprise heterogeneous hardware. I currently study the characteristics of different workloads and hardware. The goal is to provide information about potential matches between workloads and hardware under the condition of energy efficiency. Such monitoring and profiling information can then be widely used. One example is workload consolidation, where a scheduler tries to put as many workloads as possible on a few physical machines. I want to predict the power of consolidated workloads and study the compatibility of them in a heterogeneous data center setup. The data generated by such profiling and monitoring tools can then smartly extend current state-of-the-art systems like Mesos and YARN.

Keywords: power; energy; heterogeneity.

1 Introduction

Due to the environmental concerns of different energy resources and the massive power consumption of large data centers, energy efficiency becomes more and more important. Cloud computing as a whole consumes more energy than entire countries like Germany or India [4]. Usually, the running costs of data centers exceed their purchase costs [1]. Hence, reducing the energy consumption within data centers reduces their total cost.

I want to achieve reduced energy consumption in data centers by providing tools under the consideration of different aspects: (1) power estimation, (2) profiling of heterogeneous environments, and (3) analysis of co-located workloads. The basis for energy-efficient data centers is measurement and estimation of power consumption. The power consumption can be measured with a physical power meter, which for example monitors the power consumption of an entire physical machine.

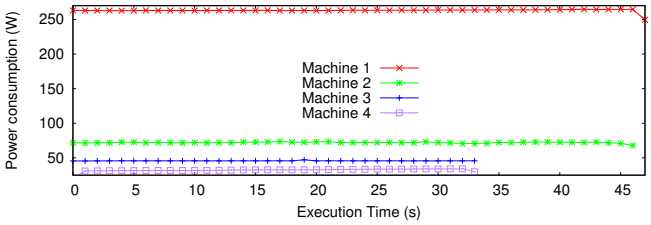


Figure 11: Power consumption of a six digit factorial computation on different machines.

This is not sufficient for fine-grained power measurements, e.g., in a virtualized environment, and software power models are needed. In a virtual machine, we do not have access to the hardware of the host. Depending on the virtualization technology, the host considers the VM as a process, and thus as a black box. In some cases it might be essential to have a process-level estimation inside a virtual machine. Depending on the type of application, multiple applications or processes are running in the same VM. As one example, in the case of Supply Chain Management applications, it is relevant to understand which component of the application consumes the most power. Hence, the first part of my work consists of process-level power estimation in virtualized environments.

Current state-of-the-art assumes that data centers consist of homogenenous hardware, but in reality this is not the case [6]. Distributed data centers for multi-cloud environments usually comprise heterogeneous hardware as they are not built at the same time by the same provider. Even in a single data center, usually different generations of hardware are in use. A specific workload consumes a different amount of energy depending on the hardware it runs on [8]. As an example, Figure 11 shows the power consumption of a six digit factorial computation on four different machines with different models of Intel CPUs. Depending on the hardware, power consumption and execution time differ. This means that, given the heterogeneity of data centers, the use of workload characteristics and profiling information of the available hardware are essential for energy efficiency. I currently study the characteristics of different workloads and hardware. The goal is to provide information about potential matches between

workloads and hardware under the condition of energy efficiency. Such monitoring and profiling information can then be widely used.

One example is workload consolidation, where a scheduler tries to put as many workloads as possible on a few physical machines. This allows to turn off the remaining machines. Here I need to study the hardware characteristics of available machines and to profile the workloads. This allows to predict the impact on power consumption when co-locating different workloads on the same physical machine. I want to study the compatibility of different types of workloads consolidated on specific hardware under the aspect of energy efficiency. The data generated by profiling and monitoring tools can then be used by systems like Mesos[2] and YARN[3].

The goal of this work is to increase the energy efficiency of data centers. Therefore, I target different aspects with my work, as depicted in Figure 12 and described in more detail in the following sections.

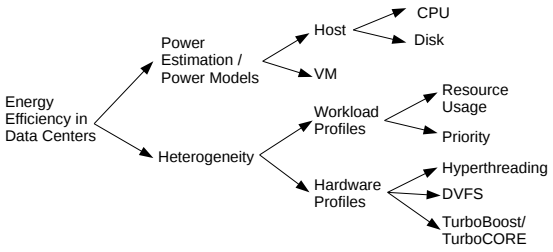


Figure 12: The different tools leading to energy efficiency in data centers.

1.1 Power Estimation

Power consumption is mainly influenced by CPU and disk utilization [9]. Thus, an estimation model covering disk and CPU power consumption can cover most workloads.

CPU. Different approaches exist to estimate the power consumption of the CPU. Current approaches make use of RAPL counters on Intel hardware [5], read performance counters (e.g., [2, 10]) or use learning techniques that assume the proportionality of system events to power consumption. With BitWatts [3] we present a fine-grained monitoring middleware providing real-time and accurate power estimation of software processes running in virtualized environments. An initial regression phase, which is done once per hardware model, allows BitWatts to build a hardware-specific power estimation model. Based on this model, a specific software process can be monitored without the need of a physical power meter. BitWatts relies on a multi-tier architecture that shares the power consumption of the VMs running on the host to the application processes running within the VM. Since the VM does not have access to the hardware of the host machine, we use a lightweight transport mechanism (VirtioSerial[4]).

Disk. Our current implementation of BitWatts covers only CPU and memory intense workloads, but not yet disk workloads. The power consumption of disk activity depends on many factors, as for example caching, journaling and disk fragmentation. Recent work shows that even hard disks of equivalent make and model can differ in their behavior [7]. Thus, the estimation of power consumption and the creation of a power model for hard disks is very challenging.

1.2 Heterogeneity

Our recent work [8] studied the energy efficiency of different workloads on heterogeneous hardware. We showed that when scheduling 5 CPU and 5 disk-intense workloads on two different machines, the scheduling has a high impact on energy efficiency. In extreme cases the energy consumption can be increased by 14 times.

To exploit those characteristics, one needs an efficient profiling of both hardware and workloads. For the workloads, I plan to define which resources they require during which period. For the hardware, I need to know the available resources and the costs for resource usage in terms of power consumption. Depending on the architecture, hardware features such as HyperThreading, DVFS and TurboBoost/TurboCORE need to be considered. To provide an efficient approach, I want to profile only a small subset of possible workload and hardware combinations.

2 Relevance and Impact

With efficient approaches for power monitoring and profiling of both workloads and heterogeneous hardware, decision making processes in data centers can be influenced towards better energy efficiency. In particular, the profiling can provide key insights with regard to workload consolidation decisions.

References

[1] L. A. Barroso. The price of performance. *Queue*, 3(7):48–53, 2005.

[2] W. L. Bircher and L. K. John. Complete system power estimation: A trickle-down approach based on performance events. In *International Symposium on Performance Analysis of Systems & Software. ISPASS*, pages 158–168. IEEE, 2007.

[2] http://mesos.apache.org

[3] http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[4] http://fedoraproject.org/wiki/Features/VirtioSerial

[3] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe. Process-level power estimation in vm-based systems. In *European Conference on Computer Systems*, EuroSys '15. (accepted), 2015.

[4] G. Cook. How clean is your cloud? Greenpeace Report, April 2012.

[5] M. Hähnel, B. Döbel, M. Völp, and H. Härtig. Measuring energy consumption for short code paths using rapl. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):13–17, 2012.

[6] T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr, and R. Bianchini. Energy conservation in heterogeneous server clusters. In *SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 186–195. ACM, 2005.

[7] E. Krevat, J. Tucek, and G. R. Ganger. Disks are like snowflakes: no two are alike. In *Conference on Hot topics in operating systems, HotOS*. USENIX Association, 2011.

[8] M. Kurpicz, A. Sobe, and P. Felber. Using power measurements as a basis for workload placement in heterogeneous multi-cloud environments. In *International Workshop on CrossCloud Systems, CrossCloud Brokers*. ACM, 2014.

[9] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys*, 46(4):47, 2014.

[10] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In *USENIX HotPower*, 2008.

CLAUDE: Cloud-computing for non-interactive long-running computationally intensive scientific applications

Andrei Lapin
Université de Neuchâtel
andrei.lapin@unine.ch

Abstract

Cloud-computing appears as a perfect environment for executing non-interactive long-running computationally intensive applications that often appear in various scientific domains. In this paper, we present CLAUDE—a cloud system that offers researchers an easy-way to execute their long running applications "in the cloud". To demonstrate the potential, we study the performance of an example scientific computing managed by CLAUDE in the virtual and bare bare metal environments.

Keywords: cloud-computing; distributed system; non-interactive application; virtual environment.

1 Introduction

Universities and research centers often maintain local High-Performance Computing (HPC) infrastructures (e.g., clusters, grids). At some point, no matter how powerful the infrastructure is, scientists may reach the limit of the available resources and experience difficulties to accommodate more data or jobs. At this point, a possible solution would require to build or search for a data center big enough to accommodate for every computing burst. However, application migration among different infrastructures is usually difficult due to significant architectural differences. Many incompatibilities may be caused by hardware or software related issues (e.g., incompatible versions). Moreover, scientific applications are often dependency-rich and may rely on many libraries. Described issues make a scientific application bound to a single computing platform, while the migration process remains a complex operation. Due to this fact, researchers are obliged to use either similar HPC infrastructures, portable programming solutions of low performance, or infrastructure-targeting solutions. We aim to solve these portability issues through a cloud-based system called CLAUDE that guarantees high performance of the execution environment and allows researchers to quickly distribute their computing load among multiple cloud centers. This work is organized in the following way.

Sec. 2 describes the benefit of cloud computing for scientific research. In Sec. 3, we present the architecture of CLAUDE—a cloud based system that facilitates scientific computing in a heterogeneous cloud environment. Sec. 4 presents the performance evaluation of example hydro-geological Monte-Carlo simulations run by CLAUDE in bare metal and various virtualization environments. Finally, we conclude in Sec. 5.

2 Cloud-Computing for science

The first important step to assure portability is the virtualization abstraction layer.

2.1 Virtualization and cloud-computing

Virtualization is a very old concept that dates back to 1974 when Popek and Goldberg derive the first set of formal requirements for efficient virtualization [2]. The idea is to run several instances of the Operating System (OS) called guests or Virtual Machines (VMs) on a single physical machine (bare metal) hereafter called the host. Virtualization is provided through a special piece of software called a hypervisor or VMM running on a physical machine. In our research, we consider two most popular open-source hypervisors, i.e., XEN[5] and Kernel-based Virtual Machine (KVM) [6], which both show the guest performance reduction of around 5% in comparison to bare metal [3].

A reasonable guest computing performance drop of around a few percent was a key to establish a service that provides VM-based resources on-demand. The computing power is provided by a cloud, which is a collection of services targeting the provision of a computing as a service paradigm. For the end-user, cloud computing can be an easy-to-use system able to cope with deployments of high complexity and having the following advantages [1]:

- **Rapid elasticity** and **scalability** via dynamic resource provisioning on a self-service basis in near real-time.
- **Measured service**, cloud resource use can be measured, controlled, and reported transparently by both the provider and consumer.
- **On-demand self service**, the consumer can acquire computer services such as applications, networks, or servers without any interaction with the service provider.

- **Broad network access** to the cloud infrastructure is available over the Internet and through standard mechanisms and protocols.
- **Resource pooling**, computing resources are pooled together to serve multiple consumers using the multiple-tenant model. Resources can be dynamically assigned and reassigned according to consumer's demands.

Virtualization allows us to separate the physical infrastructure and create dedicated virtual resources. It is a key component of the Infrastructure as a Service (IaaS) cloud, which enables tenants to receive access to VM-based computing resources. The tenant can scale resources in/out to meet the demand, for example, by releasing VMs on idle, or requesting new VMs upon a work-load burst.

2.2 Motivation, Challenges and Requirements

Due to scalability and on-demand service featuring, cloud computing offers a promising solution to the portability/performance problems described in Sec. 1 However, providing researchers with access to a cloud platform is not enough to solve these problems, while designing an efficient distributed cloud-based application requires experience of cloud technologies. In particular, the development process of a distributed application becomes tedious for researchers inexperienced in computer science. Our research is motivated by the need of designing a HPC-like cloud system that does not face the aforementioned problems and allows researchers to benefit from the advantages of the cloud infrastructure used as a typical HPC system.

We concentrate on the following aspects of our system:

Service-Oriented design. The system has to fulfill the service-oriented concept, meaning that the user is not required to maintain hardware infrastructures and only rents remote resources from a cloud provider.

Portability. Simple and quick (re-) deployments of the system on various platforms.

Heterogeneous resource utilization. Ability to combine different types of local and remote resources and use them together within a single resource pool.

Resource selection and resource necessity prediction. Appropriate resource selection mechanisms for a specific application (how to allocate tasks to cloud workers). Minimizing the VM initialization time by predicting VM requirements and proactive VM spawning (e.g., by keeping a pool of stand-by VMs ready to start computing immediately and minimizing the instantiation overhead).

General purpose computing. Ability to run various applications with no or little changes to the system core.

Additionally, we want to make sure that the cloud infrastructure is efficiently managed in the case of a common type of non-interactive long-running computationally intensive applications.

3 CLAUDE: Cloud-Based Computing-Oriented Service

This section introduces the CLAUDE system. In the following, we present architecture of the system and an example application that we use to measure the system performance.

3.1 System architecture

The system consist of three main components, i.e., *the Communication Interface*, *the Application Controller* and *the Centralized Storage*. Figure 13 illustrates the CLAUDE system architecture and inter-component information flows. *The Communication Interface* and *the Application Controller* have to be installed manually by the user. The installation process does not have any specific requirements; it can be a local or remote installation. The third component (i.e., *the Centralized Storage*) is a Cloud-Based Storage service, which is often offered by cloud providers. In this case, the user does not need to deploy his/her own *Centralized Storage* and may use a remote solution of the cloud provider. *The Communication Interface* and *the Application Controller* interact through message exchanges. We use the standardized AMQP protocol and the RabbitMQ messaging system on both sides. We defined certain types of messages to control the application life-cycle, most of the message-logic works in the request/response manner. Each component knows the type of messages it can send and receive thus contains appropriate logic for message processing. In *the Centralized Storage*, we keep input/output data of the applications executed by the system. The user can access the data through *the Communication Interface*. We use the standardized S3 protocol to interact with the

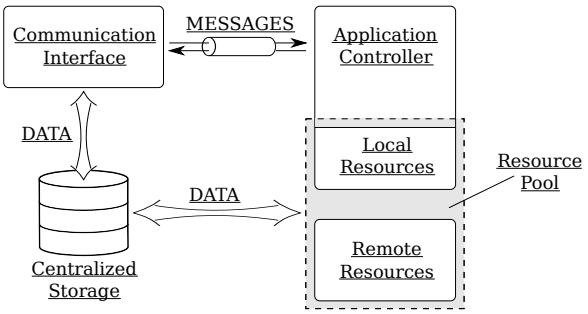


Figure 13: The CLAUDE system architecture

storage thus, it is required for *the Centralized Storage* to support this protocol. For distributing and executing applications, *the Application Controller* uses predefined resources from *the Resource Pool*. To define available resources the user has to describe it (i.e., RAM, CPU) and provide an interface through which *the Application Controller* can access the resource. *The Resource Pool* combines available local and remote resources, and *the Application Controller* chooses an appropriate resource combination for a particular application. We designed the system in a way that components remain loosely coupled and communicate only through standard protocols. The user always communicates with the system through *the Communication Interface*. One of the most convenient and intuitive implementations of *the Communication Interface* might be a Web-based application, which could allow users to provide additional meta-data information concerning application execution states or input/output data.

The core component of the system is *the Application Controller*. Figure 14 shows main subcomponents of *the Application Controller*. *The Controller*

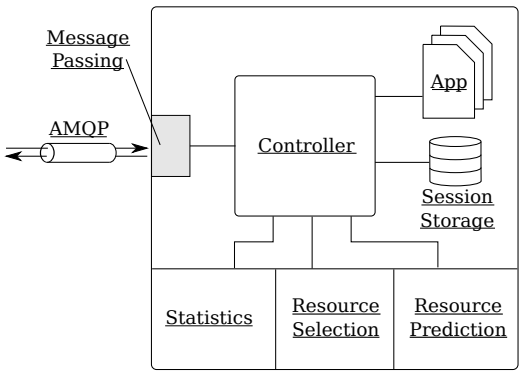


Figure 14: Application controller

takes care of the applications scheduling and maintaining correct relations between running applications and corresponding computational resources. *The Session Storage* is used for keeping backups of existing relations that the system could recover in the case of machine failure. Last three subcomponents, i.e., *the Statistics*, *the Resource Selection* and *the Resource Prediction* are mainly responsible for improving the overall system performance by preliminary VM spawning and predicting the combination of resources necessary for a particular application.

3.2 Example application

To test our system, we have chosen *HydroGeoSphere (HGS)*, which is a highly used computing application in the hydro-geological research domain that perfectly matches our initial requirements, i.e., it is a non-interactive long-running computationally intensive application. The application has been designed to solve a variety of problems with different granularity (e.g., regular geometry, steady-state saturated flow, integrated flow, solute, heat transport in regular-scale catchments). *HGS* computes physics-based simulations and uses numerical methods to solve nonlinear equations [4]. The execution time may vary from minutes to hours depending on the corresponding input and the problem complexity.

4 Evaluation

This section evaluates the performance of the chosen *HGS* application in various environments and shows the performance drop, which comes with virtualization. We focused on the most important evaluation parameter, which is in this scenario, the application execution time. In the case of our experiments, we used four different HGS input models (i.e., model_1, model_2, model_3, model_4) with different complexity of the computations. Each experiment is executed multiple times to calculate the average turn-around time and its standard deviation.

The HGS application performs two steps: the first step mostly includes the I/O operations, data preparation and memory allocation; and the second step executes actual hydro-geological problem solving. From the OS point of view, these two steps are significantly different. The first step requires many instructions to be executed in the OS privileged mode (the Kernel mode) unlike the second step, which only requires the user mode without accessing peripherals. We have performed two sets of experiments, where each set was targeted to show the performance of a particular step.

In our first set of experiments, we measured the execution time of the computationally intensive step (the second step). Indeed, a typical computationally intensive application in the virtual environment performs very close (in terms of execution time) to the bare metal execution (Figure 15). This is the result

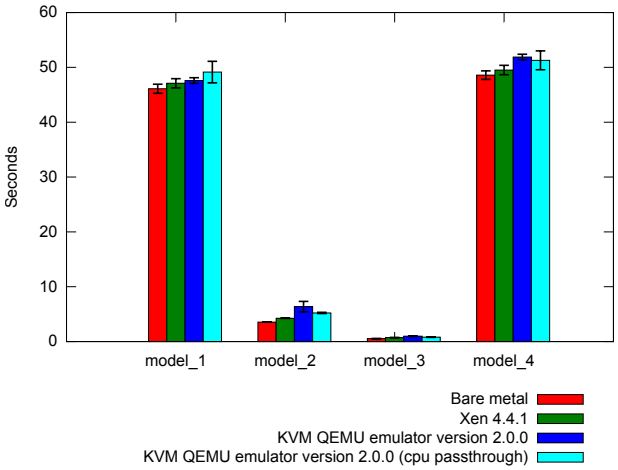


Figure 15: Time spent in user mode

we expect, because most of the instructions are executed in user mode directly on the CPU without any interaction with the hypervisor and host OS.

In our next set of experiments, we measured the execution time for the kernel mode instructions (the first step matches pretty well this requirement). The kernel mode instructions cannot be directly executed on the CPU and first have to be passed to the hypervisor and the host OS (e.g., I/O operations, memory allocation), which certainly increases the execution time. Figure 16 shows the execution time comparison in different environments. We see that the performance drop

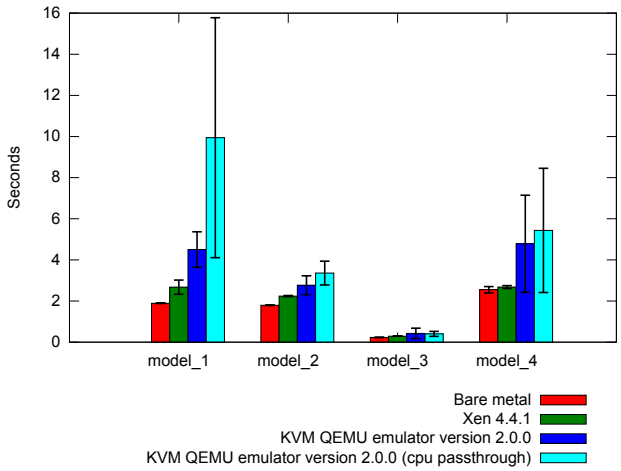


Figure 16: Time spent in the kernel mode

is more significant, however, it does not severely reduce the overall execution performance, because for our application the majority of time is always spent on the computationally intensive step (see Figure 15).

5 Conclusion

We have designed and implemented a HPC-like cloud system, which allows researchers from different scientific domains to efficiently use cloud resources for non-interactive long-running computationally intensive applications. Our system proves that the performance of virtual environments is very close to bare metal for the selected application use-case.

References

[1] Mell, Peter M. and Grance, Timothy: SP 800-145. The NIST Definition of Cloud Computing, *National Institute of Standards & Technology*, 2011

[2] Popek, Gerald J. and Goldberg, Robert P.: Formal Requirements for Virtualizable Third Generation Architectures, *ACM*, 412–421, 1974

[3] Varrette, S. and Guzek, M. and Plugaru, V. and Besseron, X. and Bouvry, P.: HPC Performance and Energy-Efficiency of Xen, KVM and VMware Hypervisors, *Computer Architecture and High Performance Computing (SBAC-PAD), 2013 25th International Symposium on*, 89–96, 2013

[4] Brunner, Philip and Simmons, Craig T.: HydroGeoSphere: A Fully Integrated, Physically Based Hydrological Model, *Blackwell Publishing Ltd*, 170–176, 2012

[5] The Xen Project: www.xenproject.org, *http://www.xenproject.org/*

[6] KVM: www.linux-kvm.org, *http://www.linux-kvm.org/*

A Passive WiFi Localization System based on Fine-grained Power-based Trilateration

Zan Li
University of Bern
li@iam.unibe.ch

Abstract

A WiFi-based passive localization system can provide user location information to third-party providers of positioning services. In this work, we provide a passive localization system for WiFi targets based on software defined radio techniques with several improved algorithms to overcome the multipath and non-line of sight problems.

1 Introduction

Typically, indoor localization techniques can be divided into range-based and range-free techniques. Range is defined as the distance between the target and an Anchor Node (AN). The first step of range-based localization, referred to as ranging, is to derive ranges to different ANs using signal parameters. Received Signal Strength (RSS) is a generic signal parameter for range-based localization and it monotonically decreases with increasing propagation distance in theory. Log Distance Path Loss (LDPL) model is the most commonly used model to relate RSS to the propagation distance. However, LDPL is not accurate for indoor localization because of multipath and Non-Line Of Sight (NLOS) propagation. Therefore, multipath and NLOS mitigation are important for range-based localization in order to achieve high accuracy.

After ranging, certain localization algorithms should be adopted to locate the target based on the range information. A typical range-based localization technique is lateration, e.g., trilateration and multilateration. Since it is still challenging to achieve highly accurate ranging, a good trilateration algorithm should be robust against ranging errors. The Linear Least Square (LLS) algorithm is a typical trilateration algorithm, which weights ranges to different ANs equally and is prone to ranging errors. The Weighted Least Square (WLS) algorithm is proposed to weight ranges to different ANs based on the range variances and has been demonstrated to be robust against ranging errors in several simulation works. However, to achieve the optimal solution, weights need the information of the ground truth ranges and the variances of each measurement, which are unrealistic in a real application.

The main contribution of this paper is a passive localization system for WiFi targets, which can extract channel information from the overheard packets to design some novel methods for ranging and positioning. The system combines (i) multipath mitigation via channel information, (ii) a novel Nonlinear Regression (NLR) method for high accuracy ranging, and (iii) a new trilateration approach, joining the Weighted Centroid (WC) and Constrained Weighted Least Square (CWLS) algorithms, to mitigate the impact of ranging errors. Key accuracy enabler in our system is the signal recovery at the physical layer, which allows us to obtain fine-grained information such as Channel State Information (CSI) and Channel Impulse Response (CIR). Special merit of the system is its ability to operate independently of the tracked device and WiFi infrastructure, i.e., passive localization with channel information, which is achieved by implementation in Software Define Radio (SDR) and not available by current off-the-shelf network cards.

2 A Passive Localization System for WiFi Signals

Figure 17 indicates the overall structure of our passive IEEE 802.11n localization system. It is comprised of three main components, signal sniffing, channel estimation, and localization algorithms. We adopt software defined radio techniques to decode the IEEE 802.11n uplink messages and extract channel information. In our work, the sniffing component is based on USRP N210 receivers that include WBX daughter-boards. Each USRP device is connected to one individual desktop, in which GNU Radio software is adopted for signal processing. After channel estimation, a server collects several useful information, e.g., timestamps, CSI, and user identities, from different ANs and runs localization algorithms to locate the targets.

3 Localization Algorithms Design

As shown in Figure 18, the localization algorithm in our system comprises of three components, which are multipath mitigation via CIR, ranging, and lateration.

3.1 Multipath Mitigation via CIR

Figure 19 indicates the amplitudes of the measured CIR in our testbed with LOS connection between the target and a USRP receiver. As shown in the figure,

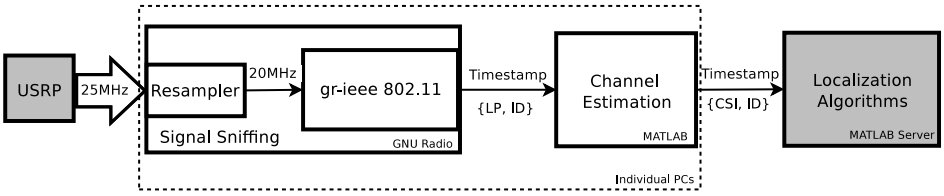


Figure 17: Passive IEEE 802.11n Localization System

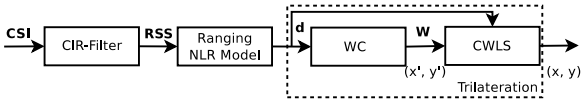


Figure 18: Structure of Localization Algorithms

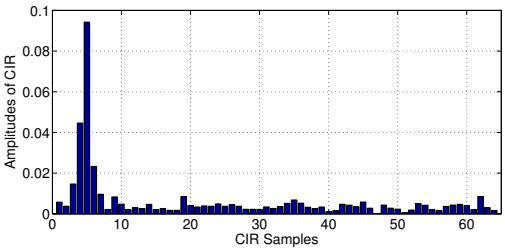


Figure 19: Channel Impulse Response

there is a path with strongest power in the CIR samples and the amplitudes in the other channels are much smaller. It is commonly known that the signal from the LOS propagation path should have the strongest power and the others are from multipath propagation. The final estimated power is as,

$$\text{RSS} = 10 \cdot \log_{10}[\max(|h(\tau)|)^2] \tag{3}$$

where $|h(\tau)|$ indicates the amplitudes of CIR over 64 samples.

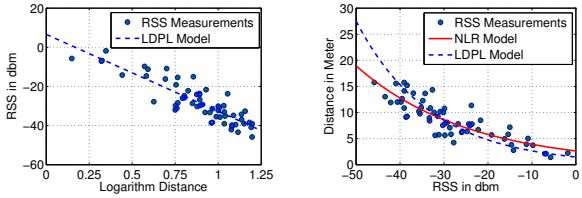
3.2 Ranging

For range-based localization algorithms, the measurement parameters, e.g., RSS, should be converted into propagation distances based on a certain model. The LDPL model is a generic model to predict the path loss for a wide range of environments. However, the LDPL model has been demonstrated to be inaccurate for indoor environments. A typical method to obtain the LDPL model is based on linear regression, which models the relationship between the RSS values and logarithmic propagation distances as a linear function as shown in Figure 20a.

In our work, we propose to model the relationship between the RSS values and propagation distances as a nonlinear curve fitting problem. Hence, we provide a nonlinear regression (NLR) model as,

$$\hat{d}_i = \alpha_i \cdot e^{\beta_i \cdot \text{RSS}_i} \tag{4}$$

where $\hat{d}_i$ is the distance between the target and ith AN, RSS_i is the RSS values obtained at the ith AN, α_i and



(a) LDPL in Log-Distance (b) NLR and LDPL

Figure 20: The LDPL and NLR Models

β_j are two unknown parameters in the model that need to be obtained from some initial measurements. In our work, we set different (α, β) pairs to different ANs.

3.3 Two-stage Trilateration Algorithm

After the ranging step, the propagation distance information are fed into a trilateration algorithm. We propose a new two-stage trilateration algorithm combining the WC and CWLS algorithms to mitigate the remaining ranging errors. The CWLS algorithm has been demonstrated as a robust trilateration algorithm for the NLOS errors. However, the typical and optimal solution to set weights based on the ground truth distances and variances is unrealistic in a practical application.

In our work, we propose to first estimate an initial location based on a weighted centroid algorithm because of its robustness to inaccurate ranging and simplicity. For the WC algorithm, the location of the target can be estimated as a weighted average of the coordinates of ANs as,

$$(x', y') = \sum_{i=1}^{M} [w_i \cdot (x_i, y_i)]. \tag{5}$$

where M is the number of ANs, (x_i, y_i) are the coordinates of the ith AN. Each weight w_i is inversely proportional to the range as,

$$w_i = \frac{\frac{1}{\hat{d}_i}}{\sum_{j=1}^{M} \frac{1}{\hat{d}_j}}. \tag{6}$$

Based on the initial location (x', y'), we can calculate the distance between the ith AN and the initial location as,

$$r_i = \sqrt{(x' - x_i)^2 + (y' - y_i)^2}. \tag{7}$$

At the second step, a suboptimal solution of CWLS algorithm is adopted following the WC algorithm to fine tune the estimated location.

For the CWLS algorithm, the location of the target can be estimated by solving a constrained optimization problem,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} (\mathbf{G}\boldsymbol{\theta} - \mathbf{h})^T \mathbf{W} (\mathbf{G}\boldsymbol{\theta} - \mathbf{h}), \tag{8}$$

subject to

$$\mathbf{q}^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{P}\boldsymbol{\theta} = 0,$$

where $\boldsymbol{\theta} = [x,\ y,\ R]^T$, $R = \sqrt{x^2 + y^2}$,

$$\mathbf{h} = \frac{1}{2}\begin{pmatrix} x_1^2 + y_1^2 - \hat{d}_1^2 \\ \vdots \\ x_M^2 + y_M^2 - \hat{d}_M^2 \end{pmatrix}, \mathbf{G} = \begin{pmatrix} x_1 & y_1 & -0.5 \\ \vdots & \vdots & \vdots \\ x_M & y_M & -0.5 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \text{ and } \mathbf{q} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}.$$

Before solving the optimal problem in Equation (8), the weight matrix $\mathbf{W}$ needs to be properly set. The optimal solution requires ground truth distance information and variances, which are unrealistic in a practical application. In our work, we set the weight matrix based on the error between the squared ranging outputs $\hat{d}_i$ and r_i in Equation (7) as,

$$\varepsilon_i = r_i^2 - \hat{d}_i^2,\ i = 1, 2, ..., M. \tag{9}$$

The covariance matrix of the disturbance is calculated as,

$$\boldsymbol{\Psi} = \text{diag}([\varepsilon_1^2, \varepsilon_2^2, ..., \varepsilon_M^2]), \tag{10}$$

and the weighted matrix can be obtained as,

$$\mathbf{W} = \boldsymbol{\Psi}^{-1}. \tag{11}$$

After setting the weight matrix, the constrained optimal problem in Equation (8) is equivalent to minimize the Lagrangian equation,

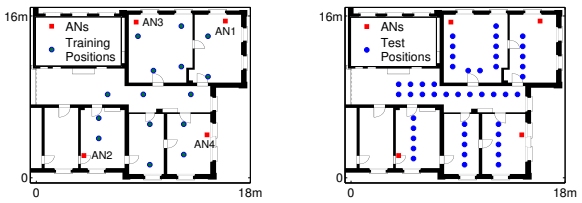
$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L(\boldsymbol{\theta}, \lambda) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} ((\mathbf{G}\boldsymbol{\theta} - \mathbf{h})^T \mathbf{W} (\mathbf{G}\boldsymbol{\theta} - \mathbf{h}) + \lambda(\mathbf{q}^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{p}\boldsymbol{\theta})), \end{aligned} \tag{12}$$

where λ is the Lagrange multiplier. To obtain the final estimation of the target location, we use the proposed method in [2] to solve the Equation (12) and to obtain the final target location (x, y) in $\hat{\boldsymbol{\theta}}$.

4 Performance Evaluation

4.1 Measurement Setup

To evaluate the localization accuracy of our proposed system, we have conducted a set of comprehensive measurements in a complex indoor environment. The system is deployed on the second floor of the IAM building in the University of Bern. Four USRP receivers are deployed in our working area as ANs to monitor the packets from a laptop as shown in Figure 21.



(a) Initial Test (b) Measurements

Figure 21: Measurement Setup

First, several initial measurements are conducted as shown in Figure 21a. 15 training positions that are spread over the whole interesting area are selected to acquire the exponential factor in the LDPL model and (α, β) in the NLR model. Second, several stationary target measurements have been conducted as shown in Figure 21b. The laptop is stationary at 46 positions, which cover the aisles in the whole target area. At each position, the data collection duration is 30s and hence around 300 packets are collected for localization.

4.2 Ranging Accuracy

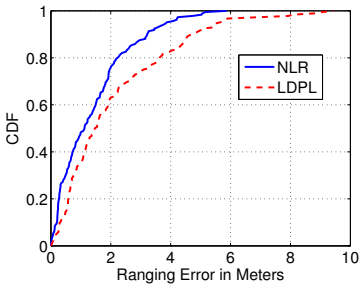


Figure 22: Ranging Errors

We calculate the ranging errors to each AN for the 46 test positions based on the NLR and LDPL models. Figure 22 indicates the Cumulative Distributed Functions (CDFs) of the ranging errors for both models. The NLR model improves the median ranging error by 27% (from $1.5m$ to $1.1m$) compared to the LDPL model. In addition, the overall improvement of the maximal ranging error based on the NLR model is about 36% (from $9.2m$ to $5.9m$). Therefore, we can conclude that the NLR model significantly outperforms the LDPL model for the ranging step.

4.3 Localization Accuracy

After evaluating the ranging accuracy, we investigate the performance of our proposed WC-CWLS algorithm with both the LDPL and NLR models.

LDPL Model: Figure 23 presents the CDFs of localization errors based on the WC-CWLS, LLS and WC algorithms.

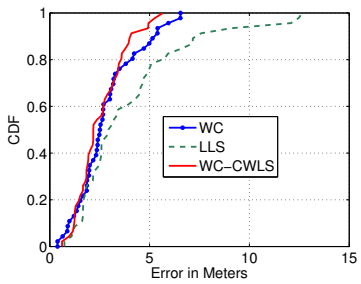


Figure 23: Localization Errors with LDPL Model

Recall that the accuracy of ranging is low with the LDPL model. Under this condition, the LLS algorithm is sensitive to the ranging errors and thus achieves the worst performance with a mean error of $3.8m$. The WC and WC-CWLS algorithms are more robust against the ranging errors, and the WC-CWLS algorithm achieves the best performance. The mean error of the WC-CWLS algorithm achieves $2.6m$, which is $0.3m$ smaller than the WC algorithm and $1.2m$ smaller than the LLS algorithm. Moreover, the maximum localization error of the WC-CWLS algorithm is $5.7m$, which is $0.9m$ smaller than the WC algorithm and $5.6m$ smaller than the LLS algorithm. CDFs of the errors in Figure 23 present a complete view on the performance of the three algorithms. We can find that the WC-CWLS algorithm generally outperforms both the WC and LLS algorithms.

NLR Model: Figure 24 shows CDFs of the localization errors for the three localization algorithms with the proposed NLR model.

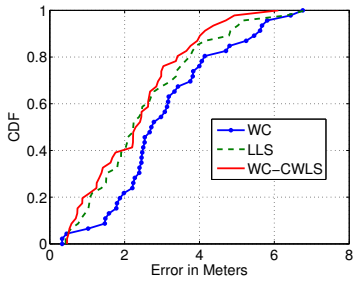


Figure 24: Localization Errors with NLR Model

Compared to the LDPL model, the NLR model improves the performance of trilateration algorithms, i.e., the WC-CWLS and LLS algorithms, due to higher ranging accuracy. The NLR model improves the mean error of WC-CWLS from $2.6m$ to $2.4m$ and LLS from $3.8m$ to $2.5m$. However, the performance of the WC algorithm becomes worse because it is in essential a proximity algorithm and its performance does not highly depend on the ranging accuracy.

With the NLR model, the WC-CWLS algorithm achieves a mean error of $2.4m$, which outperforms both the LLS ($2.5m$) and WC ($3.1m$) algorithms. With a higher ranging accuracy, the performance of the LLS algorithm gets significantly improved and is comparable to the WC-CWLS algorithm. Based on the CDF curves in Figure 24, the LLS algorithm achieves quite similar performance as the WC-CWLS algorithm for small errors (smaller than $3m$). However, the WC-CWLS algorithm is generally better than the LLS algorithm for the error larger than $3m$ because the WC-CWLS algorithm is more robust to ranging errors. With a high ranging accuracy, the WC-CWLS algorithm significantly outperforms the WC algorithm.

5 Conclusions and Future Work

We designed a software defined radio based passive localization system for WiFi signals. Channel impulse response is adopted to mitigate the influence of multipath propagation and a new ranging model is proposed based on nonlinear regression. A new trilateration algorithm combining the Weight Centroid (WC) algorithm and Constrained Weighted Least Square (CWLS) algorithm, i.e., the WC-CWLS algorithm, was designed to mitigate the influence of ranging errors. The proposed WC-CWLS algorithm is more robust to ranging errors than the LLS algorithm and achieves better localization accuracy than the WC and LLS algorithms.

In the future work, we plan to design tracking algorithms, e.g., Bayesian filters, in the system to support high accurate tracking and investigate the accuracy of the system on different moving traces.

References

[1] Z. Li, T. Braun, D.C. Dimitrova "A Passive WiFi Source Localization System based on Fine-grained Power-based Trilateration," *WoWMoM*, 2015.

[2] Cheung, K. W. and So, H. C. and Ma, W.-K. and Chan, Y. T., "A Constrained Least Squares Approach to Mobile Positioning: Algorithms and Optimality", *EURASIP J. Appl. Signal Process*, 2006.

Probabilistic Data Aggregation Protocol Based on Ant Colony Optimization in Wireless Sensor Networks

Yao Lu
Haute école d'ingénierie et d'architecture Fribourg
yao.lu@edu.hefr.ch

Abstract

In Wireless Sensor Networks (WSNs), the data aggregation techniques have the ability of reducing the data redundancy and the communication load, therefore, it contributes to the decrease of energy consumption and the prolongation of the network lifetime. The deterministic protocols pre-construct the stationary structure to perform data aggregation, however, the overhead of construction and maintenance always outweigh the benefits of data aggregation under dynamic scenarios. The probabilistic protocols dynamically make the routing decision, however, without the explicit downstream and upstream relationship of nodes, it is difficult to obtain high aggregation efficiency. In order to address these aspects, we propose a new probabilistic aggregation protocol based on Ant Colony Optimization (ACO). The Multi-Objective Steiner Tree (MOST) is defined as the optimal structure for data aggregation, which can be explored and exploited in routing process. Dynamic routing decision ensures the adaptability for topology transformation. Moreover, by using the prediction model based on sliding window for future arriving packets, the adaptive timing policy can reduce the transmission delay and enhance the aggregation probability. Therefore, the packet transmission converges from both spatial and temporal aspects for data aggregation. Finally, simulation results validate the feasibility and the high efficiency of the novel protocol when compared with other existing approaches. **Keywords:** Wireless Sensor Networks; Data Aggregation; Multi-Objective; Steiner Tree; Ant Colony Optimization; Sliding Window

1 Introduction

WSNs are widely used for perceiving the physical environment and providing the meaningful information to the observer. Since sensors are intentionally and densely deployed, the data gathering events are possible to concurrently trigger the responding actions from a portion of sensors. In the normal case, the direct transmission of data packet from source nodes to the sink node leads to high data redundancy and high communication load. Data aggregation has the ability of addressing this issue through the combination of data packets. The inherent redundancy in the raw sensor data can be eliminated, and then, the energy consumption can be reduced and implicitly, the network lifetime can be extended [1]. The existing in-network aggregation techniques can be classified as deterministic protocols and probabilistic protocols. In the deterministic protocol, the stationary aggregation structure is constructed at first. After the nodes get the knowledge of their own upstream and downstream neighbors, then the deterministic aggregation is performed on this structure. In the probabilistic protocols, the routing decision is dynamically made on the nodes. There is no explicit binding relationship between nodes, nodes do not have explicit knowledge of the downstream and upstream neighbors, and the occurrence of the aggregation is probabilistic.

The condition of probabilistic aggregation is that the aggregation can only be performed when the packets with same sequence number arrive at the same node and in the same time interval. In order to meet these requirements, an efficient routing structure should be explored and exploited for data aggregation, meanwhile, an timing policy should be adopted to control the reception and transmission of packets.

The optimal routing structure for data aggregation is described as Multi-Objective Steiner Tree (MOST), the detailed definition can be found in the previous work [2]. Four common objectives are considered for this routing structure: reducing the energy consumption, prolonging the network lifetime, reducing the aggregation delay and the communication interference. The corresponding objective functions are defined as e(x), l(x), d(x) and i(x). The optimization framework for there objectives is defined as follow:

$$\begin{aligned} &Min\ [F(x) = f(e(x), l(x), d(x), i(x))] \\ &s.t.\, g_i(x) > 0 (i = 1, 2, 3 \cdots p) \end{aligned} \tag{13}$$

where x is the feasible set of tree structure, F(x) and g_i(x) denote the fitness function and the constraint condition respectively, and p is the number of constraints. Therefore, the optimal routing tree transforms to the tree with minimum fitness value.

Without centralized scheduling protocols, the timing policy is developed to improve the performance of data aggregation on the intermediate nodes. Timers are set up on each node to control the time point of transmission, the received packets are aggregated and transmitted after the timer expiration. The configuration of timer interval needs to consider two objectives. The aggregation probability should be maximized, which means that more packets are received

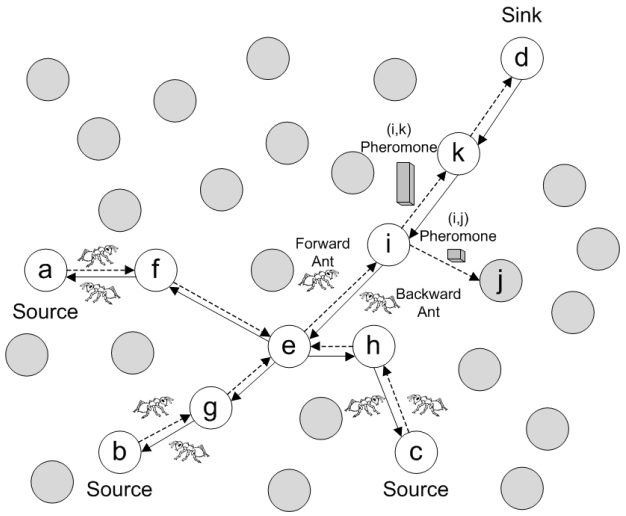


Figure 25: Routing process

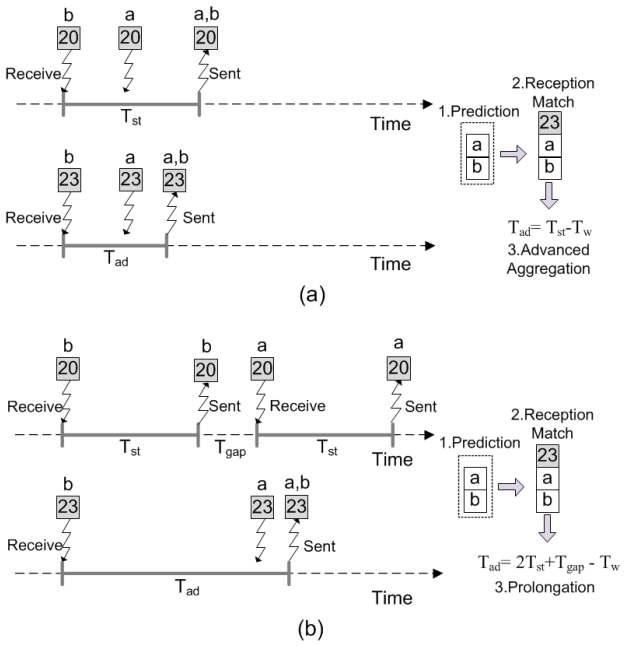


Figure 26: Adjustment of timer interval

and combined in one timer interval, and then, the normalized transmission number for each packet can be reduced to the minimum value. In addition, for the packets with same sequence number, the aggregation delay represents the time difference between the transmitting time of the first packet on source and the receiving time of the last packet on sink, and the delay value should be minimized. Suppose the aggregations are achieved on the optimal tree, the theoretical value of both objectives can be acquired by the functions relating to time interval. Let us assume that the tree has the root node at level 0 and the leaf nodes at level h. The default timer interval is T_{st}, the maximum transmission delay on layer i is denoted as $D_i(T_{st})$, and the transmission number of node j on layer i is denoted as $h_i^j(T_{st})$, where the layer i has m(i) nodes, in addition, each source generates n_{se} packets. Then the objectives of interval configuration (aggregation delay, normalized number of transmission) can be described as follows:

$$Min \left[\sum_i^h D_i(T_{st}), \frac{\sum_i^h \sum_j^m h_i^j(T_{st})}{|N_{src}| \times n_{se}}\right] \qquad (14)$$

2 Work accomplished

The probabilistic data aggregation protocol is developed to address the aggregation issue. It contains two components: routing scheme base Ant Colony Optimization(ACO) [3] and timing policy based on sliding window.

2.1 Routing Scheme

The routing scheme consist of forward, backward ant transmission, structure evaluation. ACO is embedded into the general routing process to traverse network and to explore the optimal structure of MOST, an example is depicted in Fig.1.

During the forward transmission, each forward ant is generated on source nodes and piggyback on the sensing data packets. The packet starts from the source node and move to the sink node through the routing scheme. The next hop nodes on the path is decided by the dynamic selection method. The route information on the path is recorded in the ant part of packet. Since the occurrence of aggregation is probabilistic, if multiple packets arrive at the same node during a given time interval, the aggregation function on these sensing data is performed. The ant information included in these packets are also naturally merged. During the structure evaluation, when all forward ants with the same sequence number are received, a new tree structure can be formed based on the routing paths of ants. By using the fitness function, the quality of this tree is calculated and compared against the current optimal tree which is found so far. If current solution fits better, then a new optimal tree is found and replaces the previous structure. During the backward transmission, the backward ants are generated from the sink node and moved towards to the source nodes by using the current optimal tree. Each link on this structure can be visited once during one-time backward transmission. Therefore, if the intermediate nodes on the tree receive the backward ants, the pheromone of corresponding neighbours are updated, otherwise, the pheromone is evaporated periodically. In this way, the more pheromone can be deposited on the links of the optimal structure.

2.2 Timing Policy

In the previous proposals of timing policy, the aggregation and transmission are activated by the expiration of the static timer T_{st} in exploration phase. In order to further improve aggregation performance, adaptive timer T_{ad} is proposed. Since the aggregation tree becomes stable after a while according to the convergence

property of structure exploration, the number and the source nodes of the received packet with same sequence number also becomes stable. Therefore, the packets that are supposedly received in the future become deterministic. In order to exploit this characteristic, in our policy, two types of adjustment on static timer are performed depending on different condition. There is no conflict between the prolongation and the shortening of timer interval, two adjustments cooperate with each other to improve the aggregation efficiency.

For the prediction of future packets, the source node of packet is recorded into a local data structure called sequence list, the information of packets with same sequence number are recorded under the same elements in the list. In this way, the source information of recently received packets can be exploited in order to predict the future packets. The sliding window is working on sequence list to detect the state of reception. Window match requires that the source nodes in each element should be same. The information of the future packets can be predicted based on the content in sliding window.

For the adjustment of timer interval, if the current node has already received all expected packets with one sequence number, then these packets are aggregated and transmitted immediately without waiting timer expiration. In Fig.2, an example is depicted in situation a, T_w is reduced, so aggregation delay can be shorten. Or if the timer expired, only parts of expected packets are received at this moment, then the timer will be prolonged for a period of time to wait other packets. An example is depicted in situation b, T_p is added, the aggregation probability can be increased, and the aggregation delay is also reduced.

3 Work in progress and future work

Even if the previous work has improved the performance of data aggregation, the convergence rate of this protocol is still severely affected by the network scale, and the possibility of falling into local optimal solution is relatively high. Therefore, we need to make efforts to develop the new exploration method for the optimal aggregation tree. In order to accelerate the convergence rate and avoid falling into local optimal solution, the combination of ACO and other heuristic algorithms is a possible solution.

The performance evaluation should consider more different network contexts and application scenarios. For example, if some nodes are mobile, some links are not stable, the influences of these complex and mutative network topologies should be evaluated. Or if there are many sink nodes, the feasibility of this protocol under different communication pattern should be verified.

In addition, the current simulator can not provide a practical distributed and concurrent computing infrastructure. We should consider to implement the protocol on the distributed network emulators, such as POP-C++, or the real-world testbeds of wireless sensor networks. In this cases, the authentic performance can be observed, and then the result can be more convincible.

References

[1] Fasolo, E.; Rossi, M.; Widmer, J.; Zorzi, M., "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol.14, no. 2, pp.70-87, 2007.

[2] Yao, L.; Comsa, I.S.; Kuonen, P.; Hirsbrunner, B., "Dynamic data aggregation protocol based on multiple objective tree in Wireless Sensor Networks," *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on* , pp.1-7, 2015

[3] Dorigo, M.; Gambardella, L.M., "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol.1, no.1, pp.53-66, 1997

On practical forwarding techniques for Delay Tolerant Networks

Ali Marandi
University of Bern
marandi@iam.unibe.ch

Abstract

In this paper, we discuss practical forwarding techniques for Delay Tolerant Networks (DTNs). We introduce DTNs and epidemic forwarding as the basic forwarding approach for in such networks. The major issue in epidemic forwarding is redundancy, which indeed hinders its practical applications. We describe our Bloom filter-based epidemic forwarding approach which manages the redundancy very efficiently with the aid of Bloom filters (BFs). Finally, we will discuss future directions of our work.

Keywords: Wireless communication, DTNs, Bloom filters, CCNs

1 Introduction

Traditional routing approaches for wireless networks (like AODV, DSR, and OLSR) rely on end-to-end connections between sources and destinations. More precisely, they assume that there is at least one end-to-end connected path between the source and the destination. However, we see only intermittent connectivity in many realistic scenarios like VANETs, and spontaneous networks. Meaning that there might be no connected path for long time between some pairs of nodes. In order to perform routing in such scenarios, opportunistic techniques have been proposed in the literature [1, 3, 4].

These approaches essentially rely on the store, carry and forward relaying. It is as though routing decisions in DTNs are more and more about increasing the probability that data and destination encounter and less and less about opting the best routes from sources towards destinations. The problem stems from the fact that in most realistic scenarios, nodes are not aware about the future contacts. Therefore, buffer management is of vital importance when our target is to increase the delivery ratio.

The first step of managing buffer space is to manage redundancy. In this paper, we propose Bloom filter-based epidemic forwarding, which controls the redundancy with the help of BFs. We opted for epidemic forwarding because it is the basic routing approach in DTNs. Moreover, it is the most promising approach for the scenarios in which nodes cannot predict the future contacts. The rest of the paper is organized as follows. In section II we will present epidemic forwarding. Section III gives a brief introduction to BFs. Section IV describes our Bloom filter-based epidemic forwarding approach and three strategies for Bloom filter management. Section V discusses the future directions.

2 Epidemic forwarding

Epidemic forwarding [1] is the basic technique for data dissemination in DTNs. The main idea is to increase the probability that packets encounter their destinations in intermittently connected networks. Therefore, once a source encounters a relay, it transmits to it the packets it has generated as well as the packets it is relaying for other sources. Relays also exchange their buffer content upon they encounter. Vahdat et al. showed in their seminal paper [1] that with the aid of this replication and mobility eventually packets will be delivered to their destination. Even if epidemic forwarding provides the best performance in terms of delivery ratio and delay, it utilizes the resources inefficiently. This is resulted from redundant transmissions which waste the bandwidth. Moreover, packets that had delivered to their destinations have to be removed from the buffers in order to free the space for the not yet delivered ones. In Section 4 we show that our Bloom filter-based epidemic forwarding provides efficient solutions for these problems.

3 Bloom filters

Bloom filter consists of a bit array and a number of hash functions. False negatives are impossible, but false positives are possible with controllable frequency. Given the size of BF is M, the number of inserted elements are N, and the probability of false positive is P, the number of hash functions will be determined by equation 1.

$$K = -\frac{M}{N}(lnP) \tag{15}$$

Therefore the tradeoff is between the space efficiency and the false positive probability.

Bloom filter [2] is a tool for compactly representing set's contents. It has many applications in distributed systems and communications networks. Paper [8] surveys its applications in distributed systems and networks. From among applications we can refer to checking Longest Prefix Match for IP and name lookup [6, 7]. In CCNs also BFs are useful for efficiently implementing Pending Interest Tables [5].

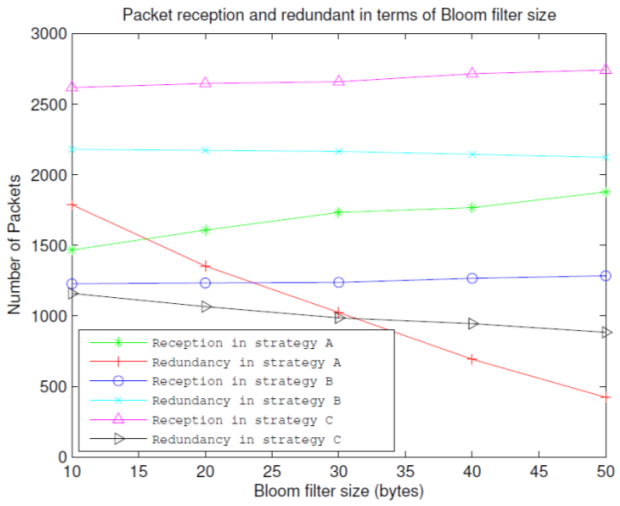


Figure 27: Reception and redundancy for different strategies with different BF overheads

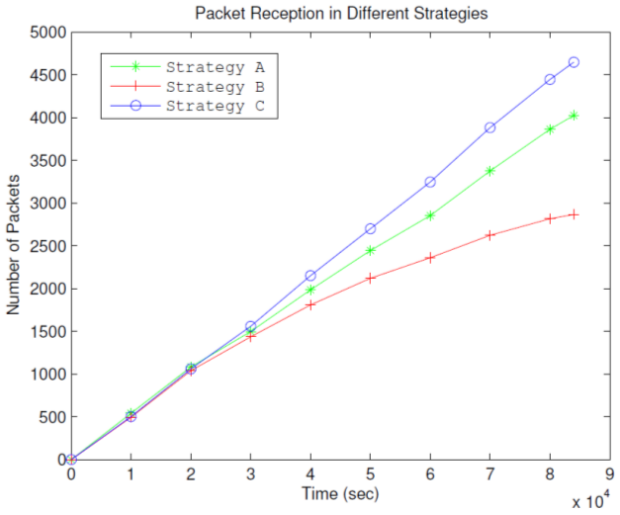


Figure 28: Packet reception for different strategies.

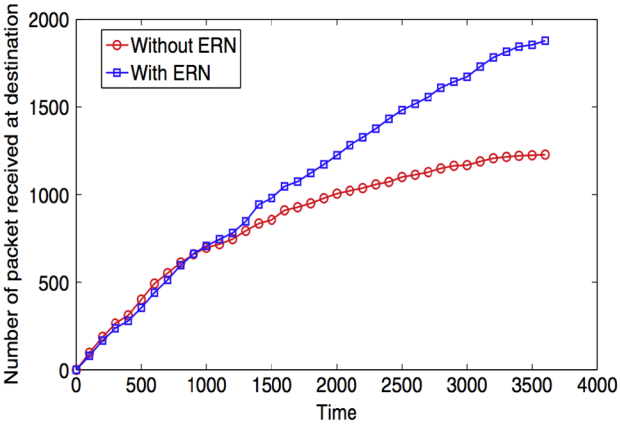


Figure 29: packet reception with and without Explicit Reception Notification

4 Bloom filter-based epidemic forwarding

We use BFs in order to compactly announce the buffer contents. In our protocol, each node inserts in a BF the packetIDs of already received packets, that is the packets in its buffer as well as the packets which have been destined to it. These BFs will be piggybacked to both packets and beacons in order to announce the buffer content to the surrounding nodes. Each node receives and maintains updated BFs from it neighbors. Therefore, it avoids sending redundant packets to them.

As time passes, the number of received packets grows. Therefore, the number of inserted elements in the BFs also increases. In order to perform BF management, we have proposed three strategies, namely strategies A, B and C. In strategy A the node inserts all recently received packets, consists of packets in the buffer and packets destined to it, in the BF. Of course, beacons can piggyback a bigger BF because they do not carry out any payloads. Threfore, in strategy B we send a big BF in beacons and a small one in the packet. Doing so, we send a bigger window of recently received packets in beacons.

Once a node decides to forward to a neighbor, it checks its buffer content in the small BF and big BF respectively. This also results in having a much smaller false positive probability equal to P^2. In strategy C, the node will put in the small BF only the packets that have been received after the transmission of the last beacon. Therefore, it has the smallest overhead compared to the two previous strategies. Fig 28 shows the number of received packets for all strategies. Fig 27 shows the number of received and redundant packets for different BF overheads.

In DTNs packets stay longer in buffers due to the lack of connectivity. Relays maintain the packets in buffers and look forward to having the opportunity of delivering them to their destinations. Therefore, it is highly important to design buffer management techniques so as to wisely evict and store packets. Such buffer management technique should evict the delivered packets from the buffers because they no longer need to be forwarded in the network. This will also free the space for other packets that are not yet delivered to their destinations. To do so, we once again utilized BFs, meaning that each beacon carries an extra BF (eBF) containing the packetIDs that have been already delivered to their destinations. Receiving the eBF, each node removes the packets represented by it from its buffer. Fig 29 shows the number of received packets with and without using eBF.

5 Future directions

We intend to extend our work to Content-Centric Networks (CCNs). Up to now, we have been assuming scenarios that the source is constant but the destination is mobile. However, in CCNs we consider the dual

scenario, meaning that the content is mobile and its location is unknown, while the content consumer is constant. With such perspective, we would like to revisit the information dissemination challenge and design efficient protocols.

References

[1] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[2] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[3] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM, 2005, pp. 252–259.

[4] ——, "Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility," in *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*. IEEE, 2007, pp. 79–85.

[5] W. You, B. Mathieu, P. Truong, J.-F. Peltier, and G. Simon, "Dipit: A distributed bloom-filter based pit table for ccn nodes," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*. IEEE, 2012, pp. 1–7.

[6] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, "Longest prefix matching using bloom filters," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 201–212.

[7] Y. Wang, T. Pan, Z. Mi, H. Dai, X. Guo, T. Zhang, B. Liu, and Q. Dong, "Namefilter: Achieving fast name lookup with low memory cost via applying two-stage bloom filters," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 95–99.

[8] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 1, pp. 131–155, 2012.

Evaluation of a Network Coding Approach for Content-Centric Networks

Jonnahtan Saltarin
University of Bern
saltarin@iam.unibe.ch

Abstract

Content-Centric Networking (CCN) naturally supports multi-path communication, as it allows the simultaneous use of multiple interfaces. To optimally use all the available interfaces, the optimal set of distribution trees should be first determined. This is not a trivial task, as it is a computationally intense procedure that should be done centrally. The need for central coordination can be removed by employing network coding, which also offers improved resiliency to errors and large throughput gains. In this extended abstract, we propose a complete architecture for integrating network coding in CCN, called NetCodCCN. The experimental evaluation of NetCodCCN shows that the proposed architecture leads to significant improvements in terms of content retrieval delay compared to the original CCN protocol. Additionally, our results demonstrate that the use of network coding adds robustness to losses.

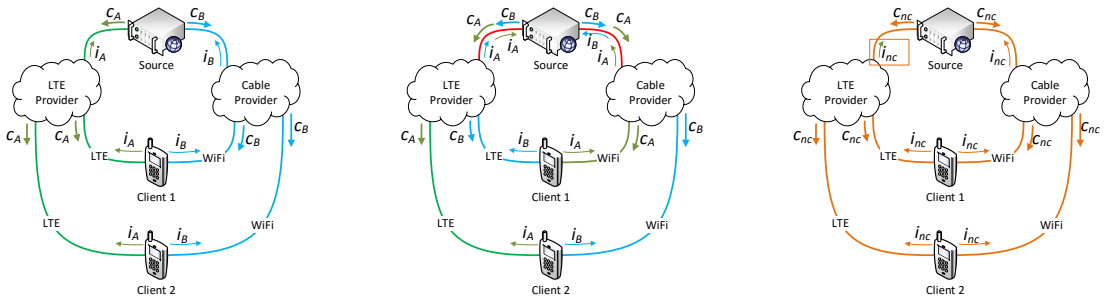
Keywords: Network Coding, Content-Centric Networks.

1 Introduction

The current Internet architecture was designed to allow communication between hosts. For this reason, in the IP protocol each packet is routed based on the location of the host which it is addressed to. However, nowadays Internet users care more about the content they want to obtain rather than where this content is stored. To address this mismatch, Jacobson *et al.* [3] proposed Content-Centric Networking (CCN), a new communication paradigm in which the importance is shifted from *where* the content is located, to *what* the content is. In the CCN model, the content is described by its *name* and the users demand content with the help of Interest messages that contain the name of the requested content. These Interests are transmitted over the network until they reach a node holding a content object whose name matches that of the Interest message. Once such a node is reached, a copy of the content object is encapsulated in a Data message and is sent back to the requester following the reverse path of that followed by the Interest message. As the Data object is transmitted backwards to the requester, intermediate nodes can store copies of it, so that they can consume future Interests for the same content.

One of the advantages of CCN is that it allows clients to exploit multiple paths in a native way. A client can simultaneously transmit Interests over all its network interfaces (*e.g.*, LTE and WiFi) to receive all the segments that comprise the requested data. This leads to a better use of the network resources and reduces the time needed to collect all the file segments. However, when multiple clients are interested in the same data file, the optimal content delivery rate is only attained if the segments are delivered over the optimal set of multicast trees [9]. See for example the Figure 30a, where both clients coordinate to use the optimal multicast trees, and Figure 30b, where the clients are not using the optimal multicast trees, and thus an additional transmission is needed to deliver the content. However, the computation of the optimal set of multicast trees necessitates a central entity that is aware of the network topology, which is hard to be done in dynamic networks. Furthermore, the transmission of Interest messages over the optimal multicast trees does not guarantee that the multicast capacity is achieved [9]. An alternative solution to the computation of the optimal multicast trees is to use network coding [1]. With network coding all the network nodes perform coding operations on the received packets instead of just replicating and forwarding them as in traditional networks. The receivers decode the information when they receive a decodable set of packets, *i.e.*, as many linearly independent coded packets as the number of source data segments. Consider the example in Figure 30c, where network coding is used, and where even when the clients do not coordinate, the best transmission performance is attained.

The application of network coding in CCN has been explored in [5] where the NC3N architecture has been introduced. Preliminary results show the benefit of this scheme. Inspired by [5], a scheme called CodingCache [8] is proposed which uses network coding to replace the data segments in the cache of the network nodes. Due to the increased segment diversity in the network, the cache hit rate is improved. Similarly to [8], in [7] a cache management framework for Information-Centric Networks with network coding is proposed. In [4], the multicast delivery of network coded data in ICN is optimized by finding the evolution of the content segments that are stored in the network. The drawback of this approach is that it does not scale well with the number of network nodes, because it needs a central entity that is aware of the network topology and the clients' requests.



(a) (b) (c)

Figure 30: Example of a multi-path transmission 30a with optimal multicast transmission, 30b without optimal multicast transmission, and 30c with network coding.

2 Work Accomplished

2.1 NetCodCCN

Motivated by the encouraging results in [5], we developed NetCodCCN, a complete architectural solution for integrating network coding in CCN. In NetCodCCN, the clients send Interest messages to request network coded data of a given name. When a source node receives an Interest for coded data, it linearly combines the original packets that match the name to create Data packets. Intermediate nodes that receive the Data packets can cache them in order to consume future requests for coded packets. The packets transmitted by the intermediate nodes are generated by linearly combining the cached coded packets. Prior to replying to an Interest with a coded packet, an intermediate node determines whether the generated coded packet will bring novel information to the receiver. If the packet has low probability to be innovative, the intermediate node forwards the Interest to other nodes. When it receives a new Data packet from the sources or another intermediate node, it generates a new coded Data packet that is then sent towards the client.

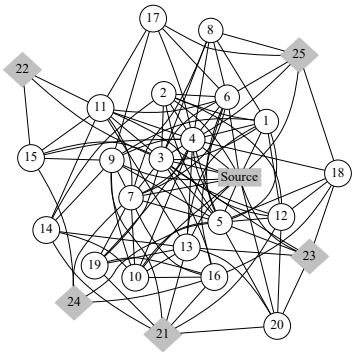


Figure 31: Planetlab topology used.

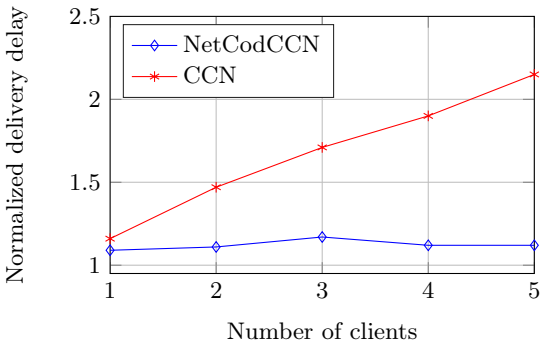


Figure 32: Normalized delivery delay versus the number of clients in the network.

2.2 Evaluation results

In this extended abstract, we will show only the results for network topologies captured by the PlanetLab project [6]. We use the network topology shown in Figure 31 that consist of one source node, 5 client nodes and 20 intermediate nodes. The links connecting the nodes have a capacity of 12Mbps.

First, we investigate how the performance is affected by the number of clients in the network. In Figure 32 we can see that with a single client, NetCodCCN and CCN perform similarly. In this case, network coding does not introduce any gains since there is only one client in the network and no losses are considered. However, the performance of CCN starts to degrade with the introduction of more clients, as they start to compete for the network resources. An optimal forwarding strategy for CCN needs to take into consideration the topology of the network in order to decide which segments each client should request over each face. However, in our work we consider that nodes are not aware of the network topology, and that there is no central entity coordinating the nodes. On the contrary, we can see that the performance of NetCodCCN does not deteriorate with the addition of new clients to the network topology. These results show that the network coding enabled NetCodCCN architecture uses more efficiently the available network resources.

We also evaluate how the error in segment transmission affects the performance of the NetCodCCN. For

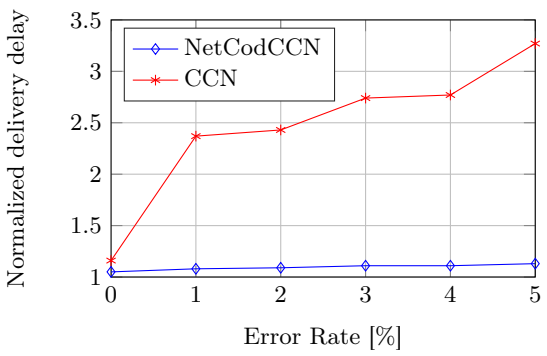


Figure 33: Normalized delivery delay versus the segment transmission error rate.

this evaluation, we choose to keep only one client, in order to compare the results with the performance of the CCN protocol. We consider losses that are caused both by the transmission losses and the errors during the processing of the segments. In Figure 33 we can see that NetCodCCN maintains the delivery delay close to the expected one, while the performance of CCN degrades very fast with the introduction of errors. This fast degradation is due to the fact that when a segment is lost, the client needs to wait until the corresponding Interest expires before it can re-send a new one.

3 Work in Progress

Currently, we are investigating the case in which the cache in the nodes is limited, and thus a decision should be made to store the most important segments. We are considering a scenario of video-on-demand transmission, where users expect low latency in order to watch the video without interruptions. We will also consider that users capabilities are heterogeneous, such that some users may prefer to view a low-resolution video, while other users may prefer high-resolution video. To address this we will consider that the video is encoded using Scalable Video Coding schemes.

4 Conclusions

We have presented NetCodCCN, a complete architectural solution for integrating network coding in CCN. In NetCodCCN, the clients express Interest messages for coded segments of a given prefix instead of asking a packet with a certain id like in CCN. The network nodes combine the Data messages by means of RLNC before forwarding them in order to take advantage of the network diversity. We have seen that the introduction of network coding in CCN eliminates the need for the clients to coordinate the transmission of Interest messages. The overall system has been tested in mesh networks with multiple clients and sources, where we have observed large performance gains in terms of the time needed to retrieve the demanded content.

References

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung. Network Information Flow. *IEEE Trans. Information Theory*, 46(4):1204–1216, Jul 2000.

[2] P. Chou and Y. Wu. Network Coding for the Internet and Wireless Networks. *IEEE Signal Processing Magazine*, 24(5):77–85, Sept 2007.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *Proc. of ACM CoNEXT*, pages 1–12, New York, NY, USA, 2009.

[4] J. Llorca, A. Tulino, K. Guan, and D. Kilper. Network-Coded Caching-Aided Multicast for Efficient Content Delivery. In *Proc. of IEEE ICC'13*, pages 3557–3562, June 2013.

[5] M.-J. Montpetit, C. Westphal, and D. Trossen. Network Coding Meets Information-Centric Networking: An Architectural Case for Information Dispersion Through Native Network Coding. In *Proc. of the 1st ACM NoM Workshop*, pages 31–36, New York, NY, USA, 2012. ACM.

[6] PlanetLab. https://www.planet-lab.org/.

[7] J. Wang, J. Ren, K. Lu, J. Wang, S. Liu, and C. Westphal. An Optimal Cache Management Framework for Information-Centric Networks with Network Coding. In *Proc. of IFIP Networking Conference*, pages 1–9, June 2014.

[8] Q. Wu, Z. Li, and G. Xie. CodingCache: Multipath-Aware CCN Cache with Network Coding. In *Proc. of the 3rd ACM ICN Workshop*, pages 41–42, New York, NY, USA, 2013. ACM.

[9] Y. Wu, P. Chou, and K. Jain. A Comparison of Network Coding and Tree Packing. In *Proc. of IEEE ISIT'04*, pages 143–, June 2004.

RESTful semantics-aware IoT enterprise integration

Matthias Thoma
University of Bern
thoma@iam.unibe.ch

Abstract

This abstract of the talk given at Doctoral Workshop on Distributed Systems, Le Louverain will give a brief overview of a service-description based interoperability framework designed for the requirements enterprise systems. It presents an Internet of Things (IoT) enterprise architecture as well as two approaches for integration of semantics at a service level: First, a top-down approach downscaling an enterprise protocol towards small embedded devices. Second, a bottom-up approach where the semantic information is encoded separately from the actual service. It features an implementation of the Constrained Application Protocol (CoAP) for IBM Mote Runner, a reactive VM-based operating system, and – on top of that – the OData protocol. Furthermore, a business operations-aware sleepy nodes implementation is presented, that allows long term sleeping of IoT-devices based upon semantic information. Evaluation results for two platforms, MEMSIC IRIS and WASPMOTE Pro, were presented.

Keywords: Internet of Things, IoT

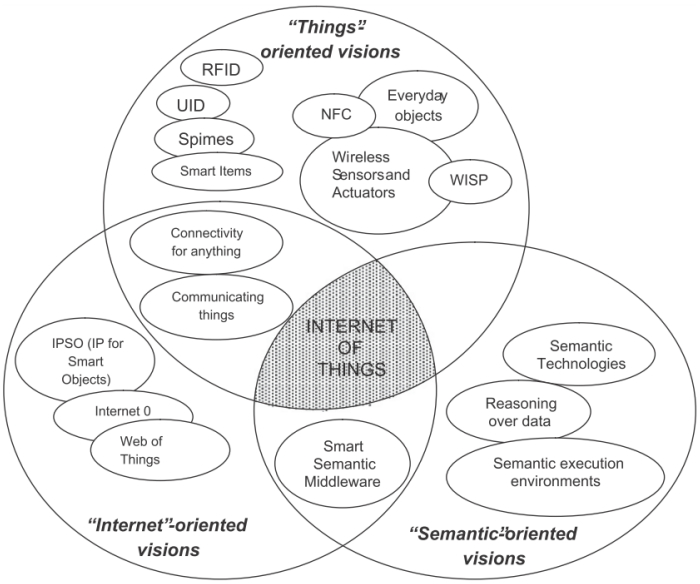


Figure 34: Converging visions leading to the Internet of Things [2]

1 Introduction

The "Internet of Things" (IoT) is the vision of a global infrastructure of networked physical objects [1]. The main idea of the IoT is the pervasive presence of things in the world and its incorporation into information systems by technologies such as sensors, RFID (Radio Frequency IDentification), actuators, and mobiles phones. According to Atzori et al. [2] the IoT paradigm can be described as the combination of the three main visions: The "Things" oriented vision, the Internet oriented vision and the semantic oriented vision. Figure 34 shows the main concepts, technologies and standards classified with reference to the vision.

The "internet-oriented" vision predicts the use of standardized internet protocols, instead of custom protocols. The "Things-oriented" vision predicts interoperating (everyday) objects based on tags, sensor and actor technologies. Last, but not least, the semantic-oriented vision is predicting the use of semantic technology for all kinds of interoperability and information description and representation. This works contributes to all three visions by presenting an Internet of Things (IoT) enterprise architecture as well as two approaches for integration of semantics at a service level: First, a top-down approach downscaling an enterprise protocol towards small embedded devices. Second, a bottom-up approach where the semantic information is encoded separately from the actual service. It features an implementation of the Constrained Application Protocol (CoAP) based on Internet technology (IPv6).

2 Service Architecture

Considering interoperability as it is currently done, one can observe that almost always a device-centric approach is used: Specific modules within the enterprise system interact with the sensing devices through proprietary protocols. Very recently the situation has changed towards IP-based protocols and standardized application level protocols.

One of the main challenges of innovation for an enterprise software vendor with a huge user base is to cope with the myriad of already existing code. Vendors aim towards innovation that has a clear integration path into already existing systems, even for more disruptive technologies. As most enterprise systems already use service repositories as an integrated part of their SOA environment, the integration of semantic service descriptions would not change the paradigm of how software is written today. Semantic Services thus could be added to enterprise software in an incremental manner without the need of disruptional changes.

Business Objects are already stored in all kinds of data stores. Here the introduction of identifiers addressing these stores are necessary. Already existing

data can be made available through service interfaces in a semantic web way. This would not introduce any changes to existing code.

More information about the service architecture can be found in [6] and [7].

3 Linked USDL4IoT

Linked USDL is one specific incarnation of the Linked Services concept. For maximizing interoperability, Linked USDL adopts, whenever possible, existing RDF(S) vocabularies. It creates explicit ontological links to domain specific ontologies. The Linked USDL vocabulary for Iot (USDL4IoT) follows the same lightweight ideas as Linked USDL does. Our vocabulary is developed in an iterative usecase driven approach. The goal of Linked USDL4IoT is to be able to publish and retrieve IoT-services in heterogeneous environments and to standardize the common core, the properties all these services have in common.

The eour principles, that guided the design of Linked USDL4IoT, are:

1. **Lightweight**: We wanted to cover the most important aspects only, avoiding to over specify and therefore limit the use of the vocabularies. In particular, domain specific knowledge needs to be specified by (external) ontologies.

2. **Coherency**: Our extension needs to integrate well into Linked USDL. Concepts and modeling style should follow the one used in Linked USDL to ensure it seamlessly integrates into existing Linked USDL vocabulary.

3. **Extensibility**: The vocabulary is designed with evolution and extensibility in mind. It needs to integrate well with existing well known external ontologies.

4. **Compatibility**: We aim for compatibility with existing widely used concepts in the enterprise field. In particular, we wanted to be able to support state of the art technologies not only RDF, but also WSDL or even proprietary sensor network protocols and data formats.

For evaluation purposes we conducted a survey [9] among researchers and industry that investigates the potential benefits and the possible pitfalls of using semantic technologies.

Linked USDL is available at http://linked-usdl.org

4 OData4IoT

The Open Data Protocol (OData)[5] is a data access protocol based on widely-used technologies (HTTP, AtomPub and JSON). Compared to the formerly predominant SOAP services, it follows a REST-based approach for a variety of data sources by defining a standardized interface. We were introducing OData into sensor networks by presenting an implementation of the OData protocol over the Constraint Application Protocol (CoAP)[4], considering both standalone scenarios as well as utilizing an intermediary. We evaluate different deployment choices and deduce recommendations for the interaction between the motes, an intermediary and an enterprise system. A comprehensive presentation of our finding can be found in [8].

5 Sleepy Nodes

Two of the main issues in wireless industrial Internet of Things applications are interoperability and network lifetime. We extended the service architecture and introduced an application-layer sleepy nodes protocol that can leverage on application knowledge in addition to energy saving measures on protocol level. A *sleepy node*, based upon [3], is defined as:

> A sleepy node is a node that may sometimes go into a sleep mode to save power and temporarily suspend all protocol communication. A sleepy node will otherwise remain in a fully powered-on state where it has the capability to perform any protocol communication. Towards the user, the sleepy node always appears as connected, thus being part of the network.

We proposed an integration platform for managing the sleep states and an application layer protocol based upon the Constraint Application Layer protocol. We evaluate our system on windowing based task allocation strategies, aiming for lower overall energy consumption that results in higher network lifetime. This work is currently under review and is about to be published in 2015.

6 Summary

The presentation gave a summary of the work done at SAP and the Communications and Distributed Systems group at the University of Bern in the field of Internet of Things and Enterprise Integration during the past four years.

We presented an Service Architecture, based upon that two integration approaches: Bottom-up with LinkedUSDL4IoT and a top-down approach where we downscaled an enterprise protocol, namely OData, to the Internet of Things. Last but not least, we showed implementation details and results of a sleepy node implementation evaluating two platforms and three possible algorithms.

References

[1] G. Kortuem, F. Kawsar, D. Fitton, V. Sundramoorthy, "Smart objects as building blocks for the internet of things" *Internet Computing, IEEE*, vol. 14, no. 1

[2] L. Atzori, A. Iera, G. Morabito, "The Internet of Things: A survey," Comput. Netw., vol. 54, number 15, pp. 2787-2805, 2010

[3] A. Rahman, "Enhanced Sleepy Node Support for Coap, Version 05", IETF draft, 2014

[4] Z. Shelby, K. Hartke, C. Bormann, Carsten, and B. Frank, "RFC 7252: The Constrained Application Protocol (CoAP)", Internet Engineering Task Force, 2014

[5] Microsoft Corporation, "Open Data Protocol (OData) Version 3.0", Specification, 2013

[6] M. Thoma, T. Braun and C. Magerkurth, "Enterprise Integration of Smart Objects using Semantic Service Descriptions", Proc. of IEEE Wireless Communication and Networking Conference (WCNC 2014), Istanbul, Turkey, 2014.

[7] M. Thoma, K. Sperner and T. Braun, "Integration of WSNs into enterprise systems based on semantic physical business entities", Proceeding of Wireless Days (WD), 2013

[8] M. Thoma, T. Kakantousis, and T. Braun, "REST-based sensor networks with OData", Wireless On-demand Network Systems and Services (WONS), 2014 11th Annual Conference on, 2014

[9] M. Thoma, T. Braun, C. Magerkurth, and F. Antonescu, "Managing things and services with semantics: A survey", etwork Operations and Management Symposium (NOMS), 2014.

Deep Learning Feature Extraction for Natural Text Analysis

Baptiste Wicht
Université de Fribourg
baptiste.wicht@unifr.ch

Abstract

Deep Learning architectures have gained a lot of support over the past years and have successfully been applied onto many fields. One of the most interesting features is that such architectures can learn features directly from raw pixels of images.

Our goal is to study such automatically-learned feature extractor and use them instead of standard features in various problem. We want to try to use these features as directly as possible, without the use of strong classifiers, in the problem of keyword spotting.

Keywords: Deep Learning, Restricted Boltzmann Machine, Convolution, Text Recognition, Keyword Spotting

1 Introduction

Deep Learning has once again become a very hot topic of the Machine Learning community. Since 2006, when Hinton introduced a new learning algorithm to efficiently initialize the weights of large neural networks, deep approaches have proved very successful in numerous fields (Text Recognition, Image Classification, Tagging, Audio classification, ...).

One very interesting characteristic of some Deep Learning approaches, such as Convolutional Deep Belief Network, is to be able to learn features directly from the pixels of an image. This avoids the need to design a feature extractor that will be used before the classification system itself. In our opinion, this makes the overall system simpler and it is an important feature of such systems.

On the other hand, the reasons why these systems are working very well is not completely understood. Moreover, features learned by these systems are generally difficult to use and needs complex classifiers or fine-tuning to achieve final results. We are looking into understanding these learned features into details and into ways of exploiting them more directly than with a complex classifier.

Section 2 presents several preliminary works that were accomplished during the course of this thesis. Section 3 presents the ongoing work of using these features for keyword spotting.

2 Work accomplished

2.1 Digit recognition in Sudokus

In this work [7], we have designed a system to detect and recognize a Sudoku puzzle on images taken from a mobile camera. Tests are made on a public dataset of 160 Sudoku images[5].

The lines of the grid are detected with a Hough transform. The grid is then recomposed from the lines. The digits position are extracted from the grid and finally, each isolated, binarized and centered character is passed to the classification system. Once each digit has been detected, the remaining task is to label each 32x32 binary image with a label from 1 to 9 (empty cells are classified a posteriori by previous steps).

A Deep Belief Network (DBN)[3] is used to recognize the digits. The network has four layers (300 hidden units for the first layer, then 300 again for the second, then 500 and finally 9 units for the final layer). Each layer is a Restricted Boltzmann Machine. The first layers are using a logistic sigmoid activation function whereas the last layer uses a simple base-e exponential. Figure 35 shows the DBN used for this task. The DBN is used as a feature extractor and a classifier at the same time. No features are extracted from the image before training.

Each RBM is first trained in an unsupervised way, one by one, using Contrastive Divergence (CD), for 20 epochs. The learning rate has been selected by experiment to be 0.1. The momentum for the first 6 epochs has been fixed to 0.5 and is then increased to 0.9 for the remaining epochs. The last layer of the network is not trained with contrastive divergence and its weights are left to values given by the Gaussian initialization.

In the second phase of training, the complete network is "fine-tuned" with the labels associated with each sample image. Conjugate Gradient (CG) Optimization has been chosen to "fine-tune" the classifier. A nonlinear Conjugate Gradient optimization method has been implemented [6]. The complete network is trained for 10 epochs.

On the test set, the complete system has an error rate of 12.5% (five Sudoku images were not perfectly recognized). This error rate is computed at the Sudoku level. If cells are considered directly, the error rate is as low as 0.37% (12 errors on 3240 cells). 58% of the errors on the cells are coming from the detector not

[5] https://github.com/wichtounet/sudoku_dataset

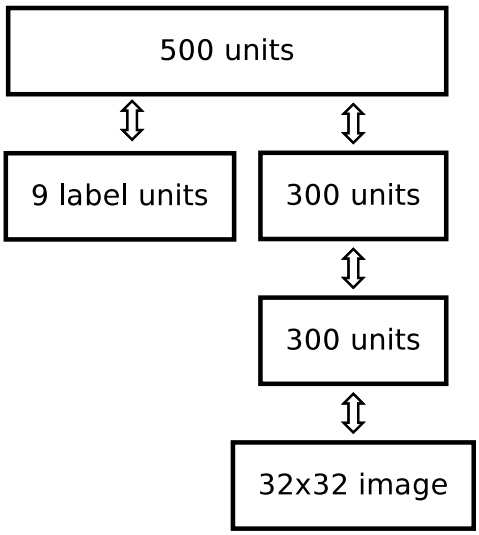


Figure 35: The network used to classify the source images with digit labels. Each layer of the network is a RBM.

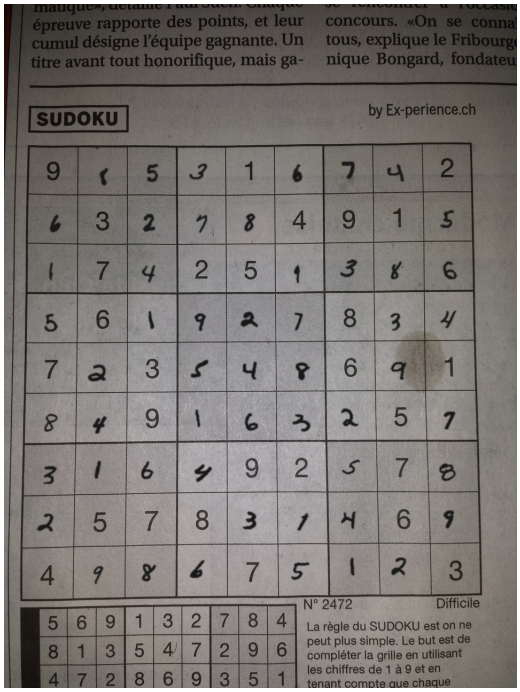


Figure 36: Image of a Sudoku puzzle from our dataset

identifying correctly an empty cell or not detecting a digit. The remaining errors are coming from wrong classification by the DBN.

Considering the poor quality of some images or the bad conditions in which some other images were taken, these results are rather satisfying. The recognition task is not very complex since all digits are computer-printed. However, the classification is highly coupled to the quality of the detection steps and to the quality of the binarization that is done on the image. As the dataset contains images that were taken in variable conditions, no thresholding method is able to perfectly binarize all the images. As such, it causes some digits to be wrongly classified by the DBN.

While training the classifier takes more than an hour, the classification process is very efficient. On average, our solution is able to produce a result from a Sudoku image in less than 100ms.

2.2 Mixed Digit recognition in Sudokus

Since computer-printed digit recognition is considered an easy problem, we have tried to increase the difficulty using Sudoku that were already filled with handwritten digits, creating Sudoku containing two types of inputs. Our goal was to see if the DBN could easily handle these two kinds of inputs without separating the inputs. Figure 36 shows an example of such an image.

The detection method remained similar to the previous one, aside from some minor tuning of heuristics. For the classifier, we opted for a Convolutional Deep Belief Network [5]. Our network was composed of two Convolutional Restricted Boltzmann Machine. Each layer used Probabilistic Max Pooling with a pooling ratio of 2. The network was not fine-tuned, instead, its features are passed to a Support Vector Machine (SVM) [1].

Each layer of the network is trained with Contrastive Divergence for 100 epochs.

A CDBN model is highly overcomplete, i.e. the size of the output representation is larger than the size of its input. The most common solution to this problem is to enforce the output representation to be sparse. The proposed system follows Lee at al. regularization method [4]. The following update (applied before weight updates) has been used during training:

$$\Delta b_k^{\text{sparsity}} = p - \frac{1}{N_h^2} \sum_{i,j} P(h_{i,j}^k = 1|v) \qquad (16)$$

Where p is the target sparsity. This update is applied to the visible biases with a specific learning rate.

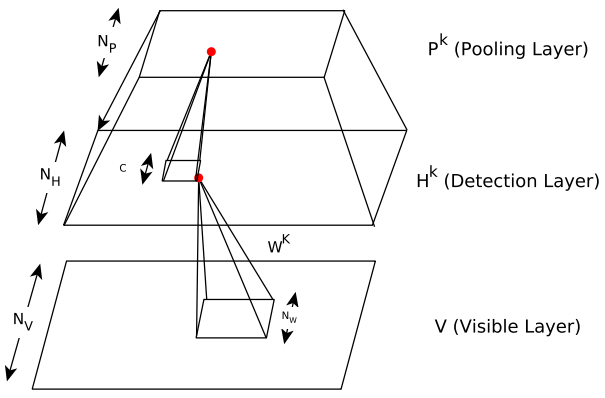


Figure 37: Convolutional RBM with max pooling

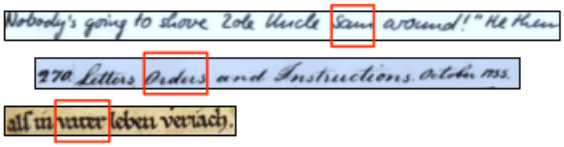


Figure 38: Keyword Spotting

The sparsity learning rate has to be chosen so that the target sparsity is reached while still allowing the reconstruction error to diminish over the epochs.

The system is thoroughly tested on a set of 200 Sudoku images captured with smartphone cameras under varying conditions, e.g. distortion and shadows. The system shows promising results with 92% of the cells correctly classified. When cell detection errors are not taken into account, the cell recognition accuracy increases to 97.7%. Interestingly, the Convolutional Deep Belief Network is able to handle the complex conditions often present on images taken with cameras and the complexity of mixed printed and handwritten digits.

2.3 Deep Learning Library

During the course of this thesis, we have developed a complete framework to experiment with Deep Learning: Deep Learning Library (DLL). This framework has been made available for download[6]. DLL is a C++ library. The library contains support for Restricted Boltzmann Machine and Deep Belief Network. It supports many types of units: binary, gaussian, ReLU or softmax. It implements Contrastive Divergence and Persistent Contrastive Divergence trainers for RBM and Stochastic Gradient Descent and Conjugate Gradient trainers for DBN. DLL also supports Convolutional RBM and Convolutional DBN. Classification with SVM is also supported.

DLL has been highly optimized for CPU performance. It has vectorized implementation of algorithms and use advanced BLAS operations to perform kernel operations. Moreover, training can be done with multiple threads. Finally, very large dataset can be handled with an out-of-memory mode.

3 Work in progress and future work

Deep Learning features have proven very efficient, but need to be used with a smart classifier such as a Support Vector Machine or fine-tuned using advanced backpropagation algorithms in order to be used for classification. Our goal is to study the features in depth and use them directly without an advanced classifier.

For this, we have selected a Keyword Spotting problem. We are using the George Washington Dataset[2] and are comparing our results against statistic features and Dynamic Time Warping. For this problem, the features generated by a Convolutional Deep Belief Network are going to be used directly without fine-tuning. The features will be compared using Euclidean Distance and images will be compared using Dynamic Time Warping.

Preliminary results indicate that this goal can be achieved, but that tuning the features, especially their sparsity and number, is of the utmost importance. Few changes in the learning parameters can dramatically change the results.

[6] `https://github.com/wichtounet/dll/`

References

[1] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. Lexicon-free handwritten word spotting using character hmms. *Pattern Recogn. Lett.*, 33(7):934–942, May 2012.

[3] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[4] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems 20*, pages 873–880. 2008.

[5] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[6] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[7] Baptiste Wicht and Jean Hennebert. Camera-based sudoku recognition with deep belief network. In *6th International Conference of Soft Computing and Pattern Recognition, SoCPaR 2014, Tunis, Tunisia, August 11-14, 2014*, pages 83–88, 2014.