An Intelligent Transportation System for Tsunamis Combining CEP, CPN and Fuzzy Logic *

Gregorio Díaz[1][0000−0002−9116−9535], Hermenegilda Macià[1][0000−0003−1462−5274], Enrique Brazález[1][0000−0001−6039−3051], Juan Boubeta-Puig[2][0000−0002−8989−7509], M. Carmen Ruiz[1][0000−0002−2392−9272], and Valentín Valero[1][0000−0003−3462−7656]

[1] School of Computer Science, Universidad de Castilla-La Mancha, Albacete, Spain
{Gregorio.Diaz, Hermenegilda.Macia, Enrique.Brazalez, MCarmen.Ruiz,Valentin.Valero}@uclm.es
[2] Department of Computer Science and Engineering, University of Cadiz, Puerto Real, Cádiz, Spain
juan.boubeta@uca.es

Abstract. Tsunamis and earthquakes have a great impact in human lives, infrastructures and economy. Although preventing tsunamis from occurring is impossible, minimizing their negative effects is in our hands. The aim of the Intelligent Transportation System (ITS) proposed in this paper is to provide safer routes for emergency and rescue vehicles. This system must consider the information regarding the tsunami alert system and the road state combined with the vehicle performance. Complex Event Processing (CEP) technology allows us to gather and process the information provided by authorities to establish the alert level. A Fuzzy Inference System (FIS) can be used to consider the uncertain regarding the road-status related concepts, such as, flood, objects and alert levels, and to assist authorities to determine whether roads are accessible. The information obtained through these technologies can then be used in a Colored Petri Net (CPN) model in order to obtain safer routes. This proposal has been applied to the Spanish city of Cádiz, due to its population density and its location in a small peninsula close to an active tectonic rift.

Keywords: Intelligent Transportation System · Tsunami · Complex Event Processing · Fuzzy Logic · Colored Petri Nets

* This work was supported in part by the Spanish Ministry of Science and Innovation and the European Union FEDER Funds under grants PID2021-122215NB-C32 and PID2021-122215NB-C33.

A seer knows best: optimized object storage shuffling for serverless analytics

Germán Telmo Eizaguirre, Universitat Rovira i Virgili, Tarragona, Spain

Serverless platforms offer high resource elasticity and pay-as-you-go billing, making them a compelling choice for data analytics. To craft a "pure" serverless solution, the common practice is to transfer intermediate data between serverless functions via serverless object storage (IBM COS; AWS S3). However, prior works have led to inconclusive results about the performance of object storage, since they have left large margin for optimization. To verify that object storage has been underrated, we design a novel shuffle manager for serverless data analytics termed Seer. Specifically, Seer dynamically chooses between two shuffle algorithms to maximize performance. The algorithm choice is based on some predictive models, and very importantly, without users having to specify intermediate data sizes at the time of the job submission. We integrate Seer with PyWren-IBM, a serverless analytics framework, and evaluate it against both serverful (e.g., Spark) and serverless systems (e.g., Google BigQuery). Our results certify that our new shuffle manager can deliver performance improvements over them.

4

STAN: Analysis of data traces using an Event-driven Interval Temporal Logic *

Laura Panizo and María-del-Mar Gallardo

ITIS Software, Andalucía Tech, Universidad de Málaga.
{laurapanizo, mdgallardo}@uma.es

Abstract. The increasing integration of systems into people's daily routines, especially smartphones, requires ensuring correctness of their functionality and even some performance requirements. Sometimes, we can only observe the interaction of the system (e.g. the smartphone) with its environment at certain time points; that is, we only have access to the data traces produced due to this interaction. This paper presents the tool STAN, which performs runtime verification on data traces that combine timestamped discrete events and sampled real-valued magnitudes. STAN uses the SPIN model checker as the underlying execution engine, and analyzes traces against properties described in the so-called Event-driven Interval Temporal Logic (eLTL) by transforming each eLTL formula into a network of concurrent automata, written in PROMELA, that monitors the trace. We present two different transformations for online and offline monitoring, respectively. Then, SPIN explores the state space of the automata network and the trace to return a verdict about the corresponding property. We use the proposal to analyze data traces obtained during mobile application testing in different network scenarios.

Keywords: Interval Temporal Logic · Runtime Verification · Trace Analysis

Related version:
This paper is published open access in the Journal *Automatic Software Engineering*, **30**,3 (2023). doi: 10.1007/s10515-022-00367-5

* This work has been supported by the Spanish Ministry of Science, Innovation and Universities project RTI2018-099777-B-I00, the Spanish Ministry of economic affairs and Digital Transformation project TSI-063000-2021-11 (5G+TACTILE-1) and the European Union Horizon 2020 research and innovation programme under grant agreements No. 101016521 (5G-EPICENTRE) and 101016608 (EVOLVED-5G)

On Data Processing through the Lenses of S3 Object Lambda

Pablo Gimeno Sarroca
Computer Engineering and Maths
Universitat Rovira i Virgili
Tarragona, Spain
pablo.gimeno@urv.cat

Marc Sánchez-Artigas
Computer Engineering and Maths
Universitat Rovira i Virgili
Tarragona, Spain
marc.sanchez@urv.cat

***Abstract*—Despite that Function-as-a-Service (FaaS) has settled down as one of the fundamental cloud programming models, it is still evolving quickly. Recently, Amazon has introduced S3 Object Lambda, which allows a user-defined function to be automatically invoked to process an object as it is being downloaded from S3. As with any new feature, careful study thereof is the key to elucidate if S3 Object Lambda, or more generally, if inline serverless data processing, is a valuable addition to the cloud. For this reason, we conduct an extensive measurement study of this novel service, in order to characterize its architecture and performance (in terms of coldstart latency, TTFB times, and more). We particularly put an eye on the streaming capabilities of this new form of function, as it may open the door to empower existing serverless systems with stream processing capacities. We discuss the pros and cons of this new capability through several workloads, concluding that S3 Object Lambda can go far beyond its original purpose and be leveraged as a building block for more complex abstractions.**

***Index Terms*—Serverless computing, object storage**

ACKNOWLEDGMENT

This research has been partly supported by EU through the Horizon Europe programme (No. 101092646, No. 101092644) and Spanish Government (No. PID2019-106774RB-22). Marc Sánchez-Artigas is a Serra Húnter Fellow. Also, Pablo Gimeno Sarroca is a Martí Franquès Research Grant Fellow — Banco Santander Ed.

Flextory: Fábrica flexible de *Consumidores de datos IoT*[*]

Rafael López Gómez, Laura Panizo y María del Mar Gallardo

ITIS Software, Andalucía Tech, Universidad de Málaga, España
{RafaelLopez, laurapanizo, mdgallardo}@uma.es

Resumen El éxito del Internet de las Cosas (Internet of Things-IoT) ha impulsado, entre otros, el desarrollo de muchas arquitecturas de software diferentes para producir, procesar y analizar datos de diversa índole. Estas arquitecturas suelen contener una aplicación principal, a la que llamamos *consumidor, encargada de las tareas de tratamiento de datos. Por lo general, los* consumidores deben construirse ad-hoc ya que algunos de sus elementos tienen que estar especialmente configurados para resolver las necesidades específicas de la aplicación que se está implementando.
En este trabajo se presenta FLEXTORY, una herramienta de fábrica de software cuyo propósito es facilitar a los desarrolladores de IoT la construcción automática de aplicaciones *consumidoras.*FLEXTORY *guía a los desarrolladores a través del proceso de generar* consumidores de Java seleccionando las funciones deseadas, como el tipo y estructura de datos que el *consumidor recibirá o el algoritmo de procesado de los datos.*

Keywords: *Fábrica Software · Internet de las Cosas · Bróker de mensajería*

1. Estado del artículo:

El artículo surgió originalmente como un Trabajo de Fin de Grado que puede encontrarse en [1].

El código se encuentra disponible en [2].

Referencias

1. https://hdl.handle.net/10630/25076, Último acceso: 2023-05-09
2. https://gitlab.com/morse-uma/formal-methods/flextory, Último acceso: 2023-05-09

[*] This work has been supported by the Spanish Ministry of Science, Innovation and Universities project RTI2018-099777-B-I00, the Spanish Ministry of economic affairs and Digital Transformation project TSI-063000-2021-11 (5G+TACTILE-1) and the European Union Horizon 2020 research and innovation programme under grant agreement No. 101016521 (5G-EPICENTRE)

Hashchain: an Efficient Setchain Implementation Built on Top of Tendermint (WIP)

Gabina Luz Bianchi[1], Margarita Capretto[1,2], Martin Ceresa[1], and César Sánchez[1]

[1] IMDEA Software Institute, Spain
[2] Universidad Politécnica de Madrid (UPM), Madrid, Spain

Abstract. A key aspect of the adoption of blockchain technologies is *their performance* [3]. Consequently, many techniques to improve the scalability of blockchains are being developed. Current blockchains require consensus algorithms to guarantee that transactions, batched as blocks, are totally ordered. Imposing a total order, although it is safe, may be unnecessary for some applications.

A promising approach to improve scalability is *Setchain*, a distributed concurrent data type that implements Byzantine tolerant distributed grown-only sets with barriers (called epochs). Setchain relaxes the total order requirement, and thus, can achieve higher throughput and scalability. Setchains can be used for those applications, like digital registries, where different elements in the blockchain need not be ordered except across infrequent barriers. Several Byzantine distributed algorithms that implement Setchain have been proposed but **no efficient real-world implementations exists**. Moreover, implementing distributed algorithms from scratch is **hard** and prone to errors.

In this work in progress [4], we propose a family of real-world Setchain implementations built on top of the *Tendermint blockchain application platform*. Our main contribution is **Haschain**, which exploits the compression power of hash functions to reduce the communication necessary during broadcasts and consensus, communicating a fixed-sized hash instead of hundreds or thousands of elements. The prize to pay is an additional distributed algorithm to obtain the set of elements from a hash, guaranteeing tolerance against Byzantine servers. We prove how Hashchain can implement Setchain correctly. We also discuss a practical implementation of Haschain built entirely on top of Tendermint, a mature platform used in several blockchain projects like Cosmos or Tezos.

Keywords: Blockchain, Setchain, Scalability, Tendermint.

[3] Mesuare in terms of the number of blocks per second.
[4] This is a preliminary draft of our ideas, we are still working on it and the document changes in a daily basis. We kindly request the reviewers and readers to understand that the presented content is subject to further refinement and improvement.

Workload-Aware Placement Strategies to Leverage Disaggregated Resources in the Datacenter

Aaron Call*†, Jordà Polo*, David Carrera*, *Member, IEEE,*
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya - BarcelonaTECH, Barcelona, Spain
E-mail: {aaron.call,jorda.polo,david.carrera}@bsc.es

I. Summary

This paper was previously published by IEEE Systems journal under DOI 10.1109/JSYST.2021.3090306 and is available at https://ieeexplore.ieee.org/document/9477130.

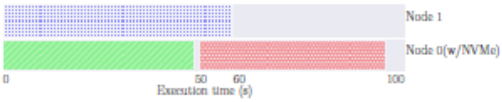
Traditional datacenter architectures comprise sets of mostly homogeneous compute platforms (also referred to as computing nodes or servers). Resources statically compose these platforms: compute, memory, storage, network fabric, and accelerators. They also access remote resources over the network, such as blob storage or network file systems. In a typical batch processing context, workloads are associated with a completion time goal or soft-deadline. They are managed by a scheduling subsystem that defines many queues or server pools used to place workloads admitted for execution.

Generally, the scheduling queues or pools provide access to a limited amount of pre-defined platform configurations, resulting in a homogeneous datacenter architecture in an attempt to make it easier to maintain and manage the infrastructure. However, homogeneous datacenters also lack adaptability, reconfigurability, malleability, and extensibility, which has an impact on the workloads.

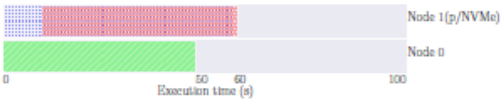
The conventional datacenter design target is to include enough infrastructure capacity to meet peak demands or to arrange the datacenter platform for bursting when needed while keeping in mind the total cost of ownership. Both of those methods depend on having enough foresight about what peak demands will be, as well as the trickier problem of predicting what possible bottlenecks might arise when applications approach capacity. Both are risky and do not account for the unexpected.

Datacenters based on Software-Defined Infrastructures (SDI) aim to disaggregate resources over a network fabric to enable increased management flexibility. On SDI, instead of pools of nodes, the pools consist of individual units of resources (storage, GPUs, FPGAs). When an application needs to be executed, SDI identifies the computational requirements and assembles all the required resources, creating a composite node. Then, the operating system is booted, and the application is instantiated. Unlike previous mechanisms to access remote resources (e.g. RDMA, or distributed filesystems), SDI resources are exposed as local to the software stack running on the composite node. Thereby, the SDI architecture virtualizes access to the pools of resources, exposing platform configurations that match the user requirements. Once the application finishes its execution, nodes are decomposed, and resources are placed back in their corresponding queues. Resource disaggregation also enables [1] increased malleability to exploit datacenter resources through sharing them across multiple workloads or by dynamically composing new ones with larger capacity or increased bandwidth.

To demonstrate the importance of disaggregation-aware workload placement policies, we will consider a simple yet illustrative example with two nodes and three workloads, as shown in Figure 1. Workload J1 requires 100% of the cores of one node (pictured green), J2 requires 50% of the cores (blue), and J3 requires one core and NVMe device (red). We consider two scenarios. In the first scenario, as shown in Figure 1a, only one node has physically-attached NVMe. When J1 and J2 arrive, they are placed in nodes 0 and 1, respectively. When J3 arrives, it only has enough cores to run in node 1. However node 1 does not have an NVMe, so J3 has to wait until J1 finishes to run on node 0. In the second scenario, in Figure 1b, the NVMe is pooled, so either node can attach and share the resource on-demand. In this case, J3 can be placed in node 1 due it can attach the disaggregated NVMe on-demand from the remote pool. Thus, disaggregation leads to more efficient use of available resources and quicker completion of jobs. While



(a) Physically-attached NVMe in node 0. Job J3 (red) has to wait until node 0 is available.



(b) Disaggregated NVMe in node 0. Job J3 (red) is executed in node 1 with NVMe pooled and remotely attached.

Fig. 1. Timeline showing the execution of three jobs in two nodes with physically-attached NVMe (top) and disaggregated NVMe (bottom)

this is a simple example of basic placement strategies, similar situations arise in many placement policies.

When the behavior of workloads under disaggregation is known, it is possible to develop policies to take maximum advantage of the resources.

Acknowledgment

This research has been partly supported by EU through the Horizon Europe programme (No. 101092644)

Análisis de Programas Concurrentes con Memoria Compartida Sensible al Contexto*

Carlos Galindo[0000−0002−3569−6218], Marisa Llorens[0000−0002−2790−0055], Sergio Pérez[0000−0002−4384−7004], and Josep Silva[0000−0001−5096−0008]

Valencian Research Institute for Artificial Intelligence (VRAIN)
Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, España
cargaji@vrain.upv.es, {mllorens,serperu,jsilva}@dsic.upv.es

Resumen La fragmentación de programas es una técnica de análisis de software cuyo uso principal es la depuración, y permite determinar qué instrucciones afectan a o son afectadas por un punto del programa seleccionado por el usuario. Para programas concurrentes con memoria compartida y múltiples procedimientos, la técnica considerada como la más precisa es la de Nanda y Ramesh, publicada en 2006. En este artículo describimos de forma detallada cómo la técnica de Nanda y Ramesh resuelve los problemas de este área, como los viajes en el tiempo, pero también cómo produce resultados imprecisos al no ser sensible al contexto en todos los casos. Para este último, aportamos un contraejemplo y detallamos la causa de esta imprecisión.

Keywords: fragmentación de programas · análisis estático · concurrencia con memoria compartida

1. Introducción

La fragmentación de programas [11] (en inglés, *program slicing*) es una técnica de análisis de software que permite extraer de un programa el subconjunto de instrucciones que afectan o son afectadas por una instrucción concreta (el criterio de fragmentación o *slicing criterion*). En sus orígenes, se concibió como una técnica de apoyo a la depuración [12] (indicando como criterio el punto en el programa en que aparece el valor erróneo), pero actualmente tiene aplicaciones diversas, desde paralelización de programas hasta detección de clones, pasando por especialización de programas y fragmentación de redes neuronales para reducir su tamaño.

* Este trabajo ha sido financiado parcialmente por la ayuda PID2019-104735RB-C41, financiada por el MCIN/AEI y por TAILOR, un proyecto financiado por el programa de investigación e innovación del Horizonte 2020 de la UE número 952215. Carlos Galindo ha sido financiado parcialmente por el Ministerio de Universidades bajo la ayuda FPU20/03861.

11

Revisiting storage for data streams: past, present, and future of the Pravega Project

Raúl Gracia , DELL

Abstract: “Nowadays, the data stream abstraction is widely used for storing and processing data both in real-time and batch. Arguably, the evolution of stream data processing has attracted the interest from researchers and the industry to a large extent, with systems like Apache Flink and Apache Spark, among others, showing rapid advancements and being backed by large companies. However, the history of the “storage side” of data streaming may not be so well known. In this talk, we will overview the different approaches that engineers and researchers have used in the last decade for persistently storing streaming data, from queueing systems to aggregators, and more recently, messaging systems such as Apache Kafka and Apache Pulsar. We will also overview why Dell Technologies started a new project in 2015, namely Pravega, to reinvent how data streams are stored and managed. We will also look forward to emerging streaming use cases that support the design decisions taken by the Pravega team when it comes to storing stream data at large scale.”