GAIA: Strengthening the Reliability of Datacenter Computing via Fast Distributed Consensus

Abstract:

To process the rapidly increasing amount of data, more and more software applications run within a datacenter containing massive computing resources. To harness these resources, applications are typically ran by either of two independent infrastructures: schedulers and virtual machines (VM). Unfortunately, as an application runs on more computers, computer errors will occur more likely at runtime and can turn down the entire application, causing disasters such as the 2015 NYSE trading halts and recent Facebook outages. Existing infrastructures lack a high availability support for applications.

This GAIA project takes a holistic methodology to greatly improve application availability via three objectives. First, we will create APUS, a fast distributed consensus protocol for general applications. Distributed consensus, a strong fault-tolerance concept, runs multiple replications of the same application and makes these replications behave consistently as long as a majority of them work normally. An open challenge is that traditional consensus protocols are slow because their protocol messages go through various software layers (e.g., OS kernels). APUS will introduce a new consensus algorithm with an ultra-fast OS kernel bypassing technique called Remote Direct Memory Access (RDMA). Preliminary results presented in [SOSP '15] show that APUS supports unmodified real-world applications, and its latest development is 32.3X to 85.8X faster than traditional consensus protocols.

Second, we will construct a first scheduler for application availability by integrating APUS with popular schedulers. A main challenge is that existing schedulers' resource allocation schemes can be conflicting with replication schemes, because the former schemes abstract away computer identity, but the later schemes must deploy replications on different computers. Our scheduler introduces a replication-aware resource allocation scheme to resolve conflicts efficiently. Our prototype scheduler presented in [APSys '16] incurs only 3.22% performance overhead on real-world applications.

Third, we will build a new VM for application availability by integrating APUS with popular VMs. One performance problem in existing VM fault-tolerance techniques is that they need to transfer all memory modified by applications across VM replications. Our new VM integrates APUS into the hypervisor layer to enforce same application inputs across VM replications, then VM replications maintain mostly same memory. We also propose a new algorithm to efficiently transfer minor different memory.

This GAIA project will greatly strengthen the reliability of many real-world applications and will benefit almost all computer users and software vendors, including many HK financial platforms. GAIA will also advance broad reliability techniques (e.g., VM migration) and infrastructures.

Long term impact:
The emergence of big data with its increasing computational demand is pushing software applications to embrace more and more computing resources. Therefore, applications now often run within a datacenter containing numerous computers. Many applications are mission-critical (e.g., financial platforms, social network platforms, and medical services), so they naturally desire both high reliability and performance.

To harness the massive computing resources within a datacenter, applications are typically ran by either of two independent infrastructures: schedulers and VMs. Depending on runtime trade-off such as performance or security isolation, some applications are ran solely by schedulers (e.g., Mesos [NSDI '11] usually uses lightweight containers, not VMs), and some other applications are ran solely by VMs (e.g., Amazon EC2).

Unfortunately, as an application runs on more computers, computer errors will occur more likely at runtime. If a failed computer happens to run an essential application component, the entire application can be turned down. A downtime often causes severe disasters, especially for online or long-running applications. For instance, due to minor computer errors, New York Stock Exchange (NYSE) was down for a whole day in 2015. Moreover, although social network applications tend to be online 24-7, minor computer errors have turned down the Facebook site for several times in recent years.

A key problem causing these disasters is that datacenter infrastructures lack a high availability support for applications. To tackle this problem, this GAIA project takes a holistic methodology: it first builds a fast consensus protocol, it then integrates this protocol into the two typical infrastructures. By doing so, applications ran by either of these infrastructures can enjoy two significant benefits.

First, the availability of mission-critical applications can be greatly strengthened. Distributed consensus is recognized a powerful fault-tolerance concept because it maintains multiple consistent replications of the same computation to overcome failures in minor replications. Although consensus consumes extra computing resources for fault-tolerance, it has been widely adopted in industry, because resource capacity is often not a bottleneck for mission-critical applications.

Second, it is easy to make fault-tolerance itself robust. Distributed consensus is notoriously difficult to understand, build, or test. For instance, although some genius companies (e.g., Microsoft) have built consensus protocols for individual applications, ironically, recent research tools have detected numerous bugs in these protocols. Building one consensus protocol for each application could be a nightmare for application developers. Fortunately, with GAIA, people only need to carefully test our protocol and infrastructures, then applications can enjoy robust fault-tolerance.

We envision significant impacts from this GAIA project in three terms.

In the near term, our fast consensus algorithm and its implementation protocol APUS (Objective 1) can drastically improve the performance of many applications that use distributed consensus. For instance, Scatter [SOSP '11], a remarkable key-value store application, deploys consensus groups using a traditional consensus protocol. By using APUS, the latency of Scatter can be one or two orders of magnitude faster.

In the intermediate term, by realizing the new infrastructures in Objective 2 and 3, GAIA will greatly strengthen the reliability of general applications and benefit almost all computer users and software vendors. For instance, HK has many financial applications which naturally demand stringent availability in operational hours, and GAIA can meet this demand. As per our tradition, we will make all GAIA source code public for research and deployments.

In the long term, we anticipate that this GAIA project will advance broad datacenter techniques (e.g., VM migration) and attract researchers to build more reliable datacenter infrastructures. As datacenter emerges to be a "giant computer", a fast and reliable datacenter OS for such a novel computer will gradually come up. Therefore, reliable consensus protocols, schedulers, and VMs will become essential datacenter OS techniques, and the outcomes of this project will be adopted in a future datacenter OS.

Overall, due to these prospective impacts, we name our project after gaia, an ancient Greek goddess who takes good care of everything on earth, including software applications.

Objectives:

1.
[To create a fast distributed consensus protocol].
We will develop an RDMA-powered distributed consensus algorithm and its implementation protocol APUS. This new algorithm aims to be one or two orders of magnitude faster than traditional consensus protocols.

2.
[To construct a first datacenter scheduler for improving application availability].
This scheduler will replicate all or essential application components by integrating APUS with popular schedulers. We will develop a new scheme to seamlessly manage both resource allocation and replication logic, so that this scheduler can efficiently schedule applications.

3.
[To build a new fault-tolerant VM for improving application availability].
We will leverage the VM hypervisor layer to transparently enforce same application inputs across VM replications. Compared to existing VM fault-tolerance techniques (e.g., primary-backup), our VM will greatly reduce the amount of transferred memory across VM replications, saving most time and network bandwidth. We will also develop a new algorithm to efficiently track and transfer minor different memory across replications.