Adding Storage Simulation Capacities to the SimGrid Toolkit

A. Lebre, A. Legrand, F. Suter, P. Veyre

Inria, Ecole des Mines de Nantes/LINA
CNRS/INRIA/University of Grenoble
IN2P3 Computing Center, CNRS

CCGrid 2015
Shenzen, China
May 06, 2015
Data is Everywhere!

Age of Data-Intensive Computational Sciences

- Data is the new source of scientific results
  - Fourth paradigm, Data deluge, Big Data, . . .
  - \( \uparrow \) Volume, \( \uparrow \) Velocity, \( \uparrow \) Variety,
    \( \uparrow \) Veracity, \( \uparrow \) Value

Storage Simulation with the SimGrid Toolkit
Data is Everywhere!

Age of Data-Intensive Computational Sciences

- Data is the new source of scientific results
  - Fourth paradigm, Data deluge, Big Data, ...
  - Volume, Velocity, Variety, Veracity, Value

Storage becomes more and more important

- Not only for historical big players
  - E.g., High Energy Physics and LHC data processing on data grids
- But in every scientific field
- And on any large scale distributed infrastructure
  - Clusters, Clouds, Grids, ...
Why Simulate Storage?

Storage: a performance driver to understand

- Independent of scale and type of the computing infrastructure
- As much important as computing and networking
- Simulation is a classical approach in performance evaluation
  - Accuracy, Scalability, and Versatility are the keys
Why Simulate Storage?

Storage: a performance driver to understand

- Independent of scale and type of the computing infrastructure
- As much important as computing and networking
- Simulation is a classical approach in performance evaluation
  - Accuracy, Scalability, and Versatility are the keys

Specifics and concerns of storage subsystems may vary

- Data Centers $\sim$ Hierarchial (mass) storage subsystems
  - Different types of media involved
- Supercomputers $\sim$ Large scale dedicated storage network
  - High-speed network interconnect
- (MapReduce) Clusters $\sim$ Specific and tuned file system
  - Reliable, scalable, and simple
- Grids and Clouds $\sim$ Set of services offered by multiple data centers
  - Hidden underlying infrastructures
Simulating Storage with SimGrid

What is SimGrid?

- 15-years old project for the simulation of distributed systems
  - but lacking of a storage component for about 10 years
- Open source, sustainable, widely used
- Available on http://simgrid.org

Main Strengths

- **Versatility**: simulates Grids, Clouds, HPC, and P2P systems
- Fast and scalable simulation kernel
- **Tractable models**: fluid models and Max-Min fairness sharing
- **(In)validation studies**: simulation results can be trusted
Simulating Storage with SimGrid

What is SimGrid?
- 15-years old project for the simulation of distributed systems
  - but lacking of a storage component for about 10 years
- Open source, sustainable, widely used
- Available on http://simgrid.org

Main Strengths
- Versatility: simulates Grids, Clouds, HPC, and P2P systems
- Fast and scalable simulation kernel
- Tractable models: fluid models and Max-Min fairness sharing
- (In)validation studies: simulation results can be trusted

Our claim
- Building a simulator from scratch should be avoided
Disclaimer

This talk is end-to-end-study-free

- Problem \(\leadsto\) Idea \(\leadsto\) Implementation \(\leadsto\) Evaluation \(\leadsto\) Problem solved!

But not contribution-free

- Comprehensive description of storage-related concepts
- Original API to develop SimGrid-based simulators
  - Leveraging a sound and reliable simulation kernel
- Performance analysis of various types of disks \(\leadsto\) Derived models
Disclaimer

This talk is end-to-end-study-free

- Problem ⇒ Idea ⇒ Implementation ⇒ Evaluation ⇒ Problem solved!

But not contribution-free

- Comprehensive description of storage-related concepts
- Original API to develop SimGrid-based simulators
  - Leveraging a sound and reliable simulation kernel
- Performance analysis of various types of disks ⇒ Derived models

Our objective

- Convince you to use our proposal to conduct your storage-related simulation studies
Outline

• Introduction

• Adding Storage to SimGrid
  Concepts and Models
  Implementation Highlights

• Checking Modeling Assumptions

• Added Value of Using SimGrid

• Conclusions and Future Work
Concepts and Models

Basic Concepts

▶ File descriptors
  ▶ Description: Name (= full path) + size [+ user-level properties]
    ▶ Remark: no UNIX info, no contents
  ▶ Life cycle: Simulated entity created by open, destroyed by close
  ▶ Local operations: open, close, read, write, seek, tell, move, and delete
  ▶ Remote operations: move and copy

▶ Storage volumes
  ▶ Description: Name + type + capacity + file list + mount point +
    attach point + simulation model
    ▶ Remark: inert file list and no navigation in tree
  ▶ Life cycle: Instantiated at parsing time
  ▶ Operations: get file list and get [total, used, available] capacity

Fluid models: Tractable and fast

▶ Assumptions (to be experimentally confirmed)
  ▶ Linearity, negligible latency, fair sharing
  \[
  \text{Maximize } \min_{a \in A} \rho_a \\
  \text{under constraints } \left\{ \sum_{a \in A \text{ using resource } r} \rho_a \leq C_r, \right\}
  \]
Implementation Highlights

Comprehensive platform description

- Scalable XML format

```
<storage_type id="SATA-II_HDD" size="500GB" content_type="txt_unix" content="unix_content.txt" model="linear">
  <model_prop id="r_bw" value="92MBps"/>
  <model_prop id="w_bw" value="62MBps"/>
</storage_type>

<storage id="Disk1" typeId="SATA-II_HDD" attach="bob">
  <model_prop id="r_bw" value="92MBps"/>
  <model_prop id="w_bw" value="62MBps"/>
</storage>

<storage id="Disk2" typeId="SATA-II_HDD" attach="alice" content_type="txt_windows" content="windows_content.txt" />

<host id="bob" power="1Gf">
  <mount id="Disk1" name="/home"/>
  <mount id="Disk2" name="/windows"/>
</host>

<host id="alice" power="1Gf">
  <mount id="Disk2" name="c:"/>
</host>

<link id="link1" bandwidth="125MBps" latency="50us"/>

<route src="bob" dst="alice" symmetrical="YES">
  <link_ctn id="link1"/>
</route>
```
Implementation Highlights

Comprehensive platform description

- Scalable XML format

Seamless remote operations

- I/O operations ↔ network transfers
  - in a store-and-forward mode

```
<storage_type id="SATA-II_HDD" size="500GB"
            content_type="txt_unix"
            content="unix_content.txt"
            model="linear">
  <model_prop id="r_bw" value="92MBps"/>
  <model_prop id="w_bw" value="62MBps"/>
</storage_type>

<storage id="Disk1" typeId="SATA-II_HDD"
         attach="bob"/>

<storage id="Disk2" typeId="SATA-II_HDD"
         attach="alice"
         content_type="txt_windows"
         content="windows_content.txt" />

<host id="bob" power="1Gf">
  <mount id="Disk1" name="/home"/>
  <mount id="Disk2" name="/windows"/>
</host>

<host id="alice" power="1Gf">
  <mount id="Disk2" name="c:"/>
</host>

$link id="link1" bandwidth="125MBps"
      latency="50us"/>

<route src="bob" dst="alice"
       symmetrical="YES">
  <link_ctn id="link1"/>
</route>
```
Outline

- Introduction

- Adding Storage to SimGrid

- Checking Modeling Assumptions
  - Experimental Setup
  - Independent Accesses
  - Concurrent Accesses

- Added Value of Using SimGrid

- Conclusions and Future Work
Experimental Setup

Testbed
▶ Grid’5000 experimental platform (http://www.grid5000.fr)

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>Interface</th>
<th>Size</th>
<th>Max. Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>griffon</td>
<td>Hitachi HDP72503</td>
<td>SATA-II</td>
<td>320 GiB</td>
<td>79 MiB/sec</td>
</tr>
<tr>
<td>granduc</td>
<td>Seagate ST9146802SS</td>
<td>SAS</td>
<td>146 GiB</td>
<td>84.7 MiB/sec</td>
</tr>
<tr>
<td>edel</td>
<td>C400-MTFDDAA</td>
<td>SATA/SSD</td>
<td>128 GiB</td>
<td>244.8 MiB/sec</td>
</tr>
</tbody>
</table>

Methodology
▶ Randomized benchmarks with FIO 2.0.8 managed with execo
  ▶ Additional dd benchmark on granduc to cope with faulty raid controller
▶ Synchronous, non-buffered I/O operations
  ▶ Independent: From 32kiB to 2GiB with a fixed block size of 32KiB
  ▶ Concurrent: 1 to 15 operations
    ▶ for 10, 50, 100, 500, 1024, and 2048 MiB files

Feel free to check and/or reproduce our results
▶ Everything is available online (http://dx.doi.org/10.6084/m9.figshare.1175156)
  ▶ Engines, raw data, analysis scripts, graphs and article sources
Modeling the Behavior of SATA-II Disks

▶ **Top:** Size vs. Duration
  ▶ Confirms the linearity assumption
  ▶ But heteroscedastic behavior
    ▶ Variability proportional to size
    ▶ Negligible latency

▶ **Middle:** Size vs. Bandwidth
  ▶ Independent of file size
  ▶ Variability $\sim$ Random variables

▶ **Bottom:** Bandwidth distribution
  ▶ Single mode but not following any well-known distribution

**Properties of derived model**

▶ Linear w.r.t. bandwidth with no latency

▶ Modeling the bandwidth-dependent variability
  ▶ Inject sample distribution and draw random variable upon access
Modeling the Behavior of SSD Disks

- **Top:** Size vs. Duration
  - Linear with very little variability

- **Middle:** Size vs. Bandwidth
  - Far from `hdparm` results
  - Default `ext4` config prevents getting maximum performance

- **Bottom:** Bandwidth distribution
  - Regular but not following any well-known distribution

Properties of derived model (similar to SATA-II)

- Linear w.r.t. bandwidth with no latency
- Modeling the bandwidth-dependent variability
  - Inject sample distribution and draw random variable upon access
Modeling Concurrent Accesses

Performance improvements on SSD

- Significant and non-linear for reads
- When having more than one write
  - Likely because of bad ext4 setup

On SAS and SATA-II

- Fixed bandwidth for writes
- Linear decay for reads
  - Explained by arm movements

Properties of derived model

- Modify resource capacity as concurrency increases
- Reevaluation each time a transfer begins or ends
- Easy to implement in SimGrid’s kernel
Outline

- Introduction
- Adding Storage to SimGrid
- Checking Modeling Assumptions
- Added Value of Using SimGrid
  - Build (and Trust) your own Simulator
  - Design (and Plug) your own Storage Model
- Conclusions and Future Work
Build (and Trust) your own Simulator

Rationale

▶ Developing a full DES from scratch is counterproductive!
  ▶ Already there: open, fast, and scalable kernel
▶ Better focus on the applicative part of the simulator
  ▶ With confidence on lower layers: (in)validated and reliable models
▶ Leverage versatility
  ▶ Mixing concepts ≠ stacking features

Examples of added value

▶ Versatility ~ Study more performance drivers w/o oversimplification
  ▶ Storage study + network interconnect + CPU heterogeneity
▶ (In)validation studies ~ get realistic results, not just some results
  ▶ Leverage predictive value in performance studies
▶ Scalable does not necessarily means inaccurate
  ▶ Both can be obtained simultaneously
Design (and Plug) your own Storage Model

There is more than disks to model

- Tape libraries → Access time (arm movements) + I/O time
  - Combination of models
- Parallel/Distributed File Systems → Disks + management layer
  - File system simulator + disks models
  - Model experienced throughput
- Storage on unknown infrastructures (Clouds) → Black boxes
  - Model with bandwidth vs. #requests matrices

How to design and plug a new model?

- Designing and plugging a fluid model is pretty straightforward
  - Behavior for a single operation + Sharing policy
- Instantiation is more complex (yet crucial)
  - Benchmarking and analysis procedures available online
- Contributions are welcomed!
Conclusion and Future Work

Conclusions

▶ Comprehensive description of storage-related concepts
▶ Original API to develop SimGrid-based simulators
  ▶ Leveraging a sound and reliable simulation kernel
▶ Thorough Performance analysis of various types of disks
  ▶ Derived Fluid models \(\sim\) tractable, fast, and accurate
▶ Only a first step . . .
Conclusion and Future Work

Conclusions

▷ Comprehensive description of storage-related concepts
▷ Original API to develop SimGrid-based simulators
  - Leveraging a sound and reliable simulation kernel
▷ Thorough Performance analysis of various types of disks
  - Derived Fluid models \(\sim\) tractable, fast, and accurate
▷ Only a first step . . .

Future Work

▷ Extend API to handle block storage, handle cache policies
▷ Integrate other resource models
  - Only after thorough (in)validation studies
▷ Study other performance metrics (e.g., energy consumption)
▷ Welcome contributions from external users
  - Now I hope you are convinced to use SimGrid 😊