Wrekavoc: An Emulator of Heterogeneity

Emmanuel Jeannot
INRIA - LORIA
Island-Lake, Flat Rock
CCGSC 2008
The discipline of computing: an experimental science

Studied objects (hardware, programs, data, protocols, algorithms, network): more and more complex.

Modern infrastructures:

- Processors have very nice features
  - Cache
  - Hyperthreading
  - Multi-core
- Operating system impacts the performance (process scheduling, socket implementation, etc.)
- The runtime environment plays a role (MPICH ≠ OPENMPI)
- Middleware have an impact (Globus ≠ GridSolve)
- Various parallel architectures that can be:
  - Heterogeneous
  - Hierarchical
  - Distributed
  - Dynamic
Purely analytical (math) models:

- Demonstration of properties (theorem)
- Models need to be tractable: over-simplification?
- Good to understand the basic of the problem
- Most of the time ones still perform a experiments (at least for comparison)

For a practical impact: analytic study not always possible or not sufficient
Experimental culture not comparable with other science

Different studies:
- In the 90’s: between 40% and 50% of CS ACM papers requiring experimental validation had none (15% in optical engineering) [Lukovicz et al.]
- “Too many articles have no experimental validation” [Zelkowitz and Wallace 98]: 612 articles published by IEEE.
- Quantitatively more experiments with times

Computer science not at the same level than some other sciences:
- Nobody redo experiments (no funding).
- Lack of tool and methodologies.
Two types of experiments

• Test and compare:
  1. Model validation (comparing models with reality)
  2. Quantitative validation (measuring performance)

• Can occur at the same time. Ex. validation of the implementation of an algorithm:
  • grounding modeling is precise
  • design is correct
Experimental validation

A good alternative to analytical validation:
- Provides a comparison between algorithms or programs
- Provides a validation of the model or helps to define the validity domain of the model

Several methodologies:
- **Simulation** (SimGrid, GridSIM, NS, …)
- **Emulation** (MicroGrid, Wrekavoc, …)
- **Real-scale** (Grid’5000, PlanetLab, DAS-3…)
Emulation: executing a real application on a model of the environment

Two approaches:
- Sandbox/virtual machine: confined execution on (a) real machine(s). syscall catch. Ex: MicroGrid
- Degradation of the environment (to make it heterogeneous): direct execution. Ex: Wrekavoc
[Chien et al. 02]:

- Real application run on virtualized resources.
- Emulation of CPU using (sandbox/virtual machines)
- Process wrapping: the same real node can execute several (VM)
- Syscall are intercepted and interpreted.
- Network: packet-level simulator (MaSSF)
- Synchronization of real and virtual time

😊 No recent dev (dec 2004).
A grid is an heterogeneous distributed environment

Goal: experiment distributed algorithm on a cluster (homogeneous and centralized)

How: transform this cluster into heterogeneous environment and control the heterogeneity
Goal

Make a cluster an heterogeneous environment

- CPU speed.
- Mémory.
- Network bandwidth.
- Network latency.

A real node

= 

An emulated node.

Two solutions:

1. Increase the performance (update the hardware)
2. Degrade the performance (by software means)

Solution 2: Costless and allows for performance control.
We want to degrade:

- CPU speed
- Allocatable Memory
- Network bandwidth
- Network latency
3 approaches :

- **CPU-freq** (Linux kernel module that change the CPU frequency)
  - Advantage: very precise.
  - Drawback: Requires ACPI enabled CPU+few usable frequencies (coarse management).

- **CPU-burning** (A process take some CPU cycle)
  - Advantage: works on any architecture + fine management
  - Drawback: calibrating is hard, degrades net. perf. to the same proportion

- **CPU-scheduling** (a user-level scheduler suspends or activates process execution according to the desired degradation).
  - Advantage: very precise (default method)
  - Drawback: uses /proc (not portable)
mlock and munlock: pins memory pages to physical memory to limit the available memory
Network management

We use (Traffic Control) of iproute2:

- Limit ingoing and outgoing bandwidth
- Limit latency (ver. 2.6.8.1 or better).
- Traffic control depends on IP addresses
Logical architecture

- The cluster is decomposed into islets.

- 1 islet = union of IP addresses intervals:
  - [152.81.2.12-152.81.2.25] - [152.81.2.151-152.81.2.176]

- Each node of a given islet shares the same characteristics

- Network characteristics are define between and inside an islet

- Iter-islet routing
### Comparing different experimental methodologies

<table>
<thead>
<tr>
<th></th>
<th>Simulation</th>
<th>Sandbox/VM</th>
<th>Wrekavoc</th>
<th>Real-scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real application</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Abstraction</td>
<td>Very High</td>
<td>High</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Execution time</td>
<td>Speed-up</td>
<td>Slow-down</td>
<td>Preserved</td>
<td>Preserved</td>
</tr>
<tr>
<td>CPU folding</td>
<td>Mandatory (?)</td>
<td>Possible</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>Controllable</td>
<td>Yes/No</td>
<td>Controllable</td>
<td>No</td>
</tr>
</tbody>
</table>
Experiments

Several experiments on Grid’5000:

- Configuration time
- Micro-benchmark
- Impact of CPU degradation against available bandwidth
- Realism of Wrekavoc
Micro-benchmark

<table>
<thead>
<tr>
<th>Set latency</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT</td>
<td>2.12</td>
<td>10.05</td>
<td>20.12</td>
<td>100.06</td>
<td>200.2</td>
<td>1000.05</td>
<td>1999.75</td>
</tr>
</tbody>
</table>
CPU Degradation vs. Bandwidth Degradation

- **Measured bandwidth (Mb/s)**
- **Desired CPU frequency**

The graph shows the relationship between the measured bandwidth (Mb/s) and the desired CPU frequency. As the desired CPU frequency increases, the measured bandwidth initially increases rapidly but then plateaus at higher frequencies.
## The Realism of Wrekavoc

Description of the heterogeneous environment

<table>
<thead>
<tr>
<th>ID</th>
<th>Proc</th>
<th>RAM (MiB)</th>
<th>System</th>
<th>Freq (MHz)</th>
<th>HDD type</th>
<th>HDD (GiB)</th>
<th>Network card (Mbit/s)</th>
<th>MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P. IV</td>
<td>256</td>
<td>Debian 2.6.18-4-686</td>
<td>1695</td>
<td>IDE</td>
<td>20</td>
<td>100</td>
<td>3393</td>
</tr>
<tr>
<td>2</td>
<td>P. IV</td>
<td>512</td>
<td>Debian 2.6.18-4-686</td>
<td>2794</td>
<td>IDE</td>
<td>40</td>
<td>1000</td>
<td>5590</td>
</tr>
<tr>
<td>3</td>
<td>P. IV</td>
<td>512</td>
<td>Debian 2.6.18-4-686</td>
<td>2794</td>
<td>IDE</td>
<td>40</td>
<td>1000</td>
<td>5590</td>
</tr>
<tr>
<td>4</td>
<td>P. III</td>
<td>512</td>
<td>Debian 2.6.18-4-686</td>
<td>864</td>
<td>IDE</td>
<td>12</td>
<td>100</td>
<td>1729</td>
</tr>
<tr>
<td>5</td>
<td>P. III</td>
<td>128</td>
<td>Debian 2.6.18-4-686</td>
<td>996</td>
<td>IDE</td>
<td>20</td>
<td>100</td>
<td>1995</td>
</tr>
<tr>
<td>6</td>
<td>P. III</td>
<td>1024</td>
<td>Debian 2.6.18-4-686</td>
<td>498</td>
<td>SCSI</td>
<td>8</td>
<td>1000</td>
<td>997</td>
</tr>
<tr>
<td>7</td>
<td>P. II</td>
<td>128</td>
<td>Debian 2.6.18-4-686</td>
<td>299</td>
<td>SCSI</td>
<td>4</td>
<td>1000</td>
<td>599</td>
</tr>
<tr>
<td>8</td>
<td>P. II</td>
<td>128</td>
<td>Debian 2.6.18-4-686</td>
<td>299</td>
<td>SCSI</td>
<td>4</td>
<td>100</td>
<td>599</td>
</tr>
<tr>
<td>9</td>
<td>P. II</td>
<td>128</td>
<td>Debian 2.6.18-4-686</td>
<td>298</td>
<td>SCSI</td>
<td>4</td>
<td>100</td>
<td>596</td>
</tr>
<tr>
<td>10</td>
<td>P. II</td>
<td>64</td>
<td>Debian 2.6.18-4-686</td>
<td>398</td>
<td>IDE</td>
<td>20</td>
<td>100</td>
<td>798</td>
</tr>
<tr>
<td>11</td>
<td>P. IV</td>
<td>512</td>
<td>Debian 2.6.18-4-686</td>
<td>2593</td>
<td>IDE</td>
<td>40</td>
<td>1000</td>
<td>5191</td>
</tr>
<tr>
<td>12</td>
<td>Dual Opteron 240</td>
<td>2048</td>
<td>Debian 2.6.18-4-amd64</td>
<td>1604</td>
<td>IDE</td>
<td>22</td>
<td>1000</td>
<td>3211 and 3207</td>
</tr>
</tbody>
</table>
- Gerbessiotis' and Valiant's sample sort algorithm
- Assume heterogeneous env.: Same load on each node
Static load balancing

Matrix-multiplication on heterogenous env. [Lastovetsky et al. 2004]
Dynamic load balancing

Advection diffusion program (kinetic chemistry)
Iterative computation (load exchange at each iteration)
Master-worker

Run the same parallel rendering using *Povray* on a heterogeneous cluster (bottom) and on an emulated one with Wrekavoc (top), (Joint work with O. Dubuisson)
Conclusion

- Computer science is also an experimental science
- Need for tools to test, compare and validate proposed solutions
- Wrekavoc an Heterogeneity Emulator
- Asses its performance and realism

http://wrekavoc.gforge.inria.fr/