Wrekavoc: An Emulator of Heterogeneity

Emmanuel Jeannot

INRIA - LORIA

Island-Lake, Flat Rock

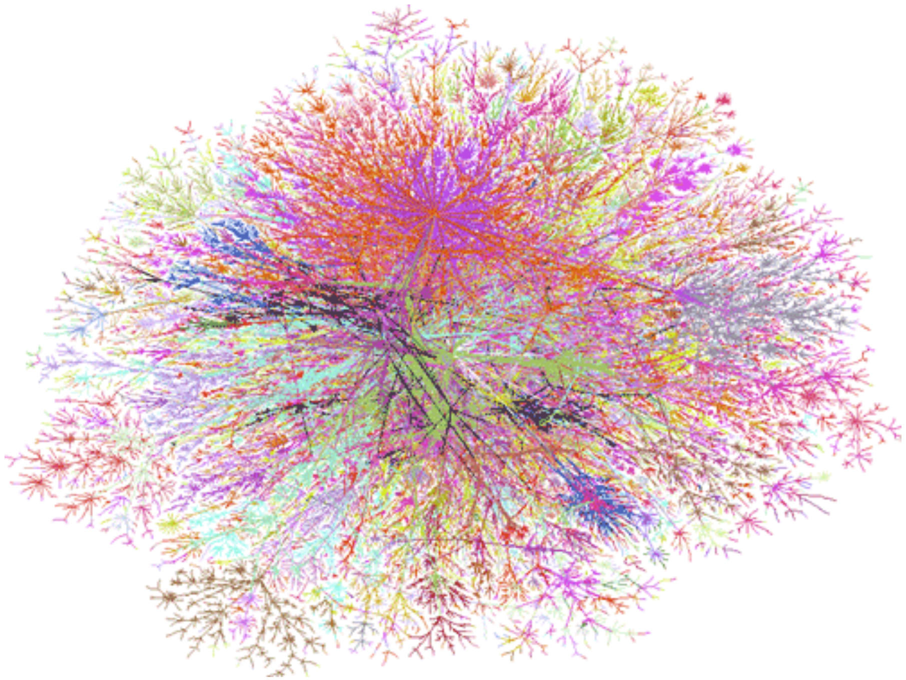
CCGSC 2008

The discipline of computing: an experimental science

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

Modern infrastructures:

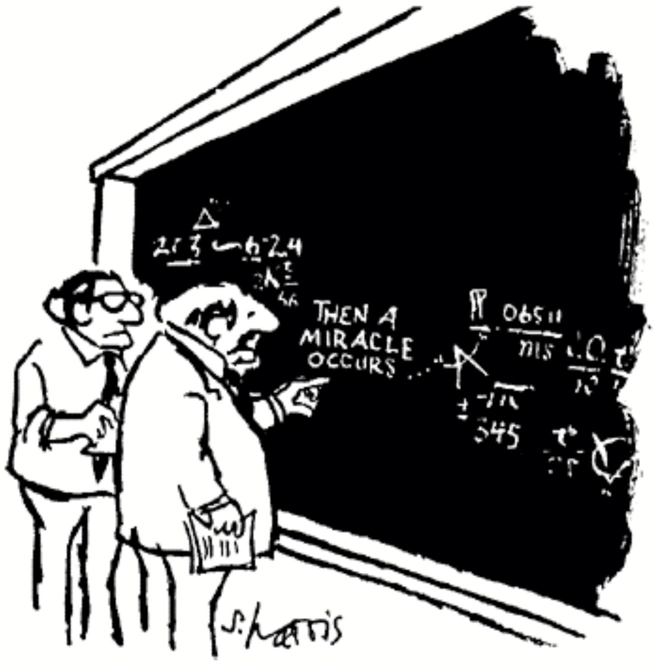
- Processors have very nice features
  - Cache
  - Hyperthreading
  - Multi-core
- Operating system impacts the performance (process scheduling, socket implementation, etc.)
- The runtime environment plays a role (MPICH≠OPENMPI)
- Middleware have an impact (Globus≠GridSolve)
- Various parallel architectures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



Analytic study

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)



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

For a practical impact: analytic study not always possible or not sufficient

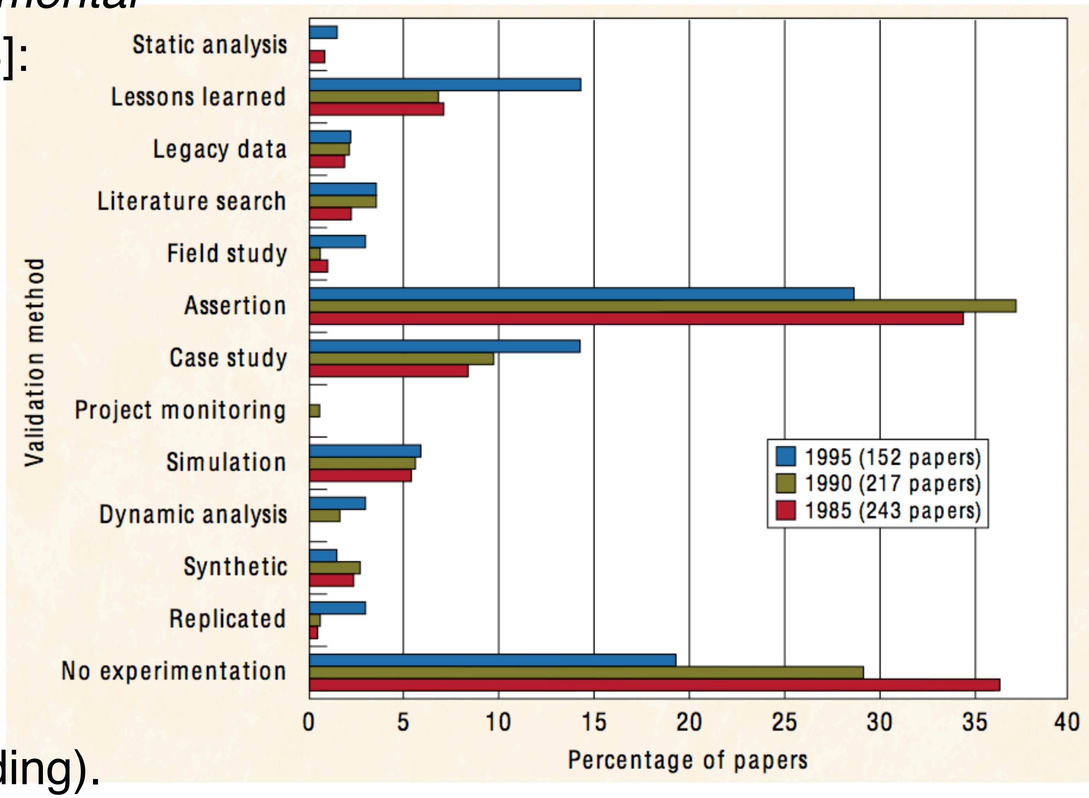
Experimental culture not comparable with other science

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



M.V. Zelkowitz and D.R. Wallace. Experimental models for validating technology. Computer, 31(5):23-31, May 1998.

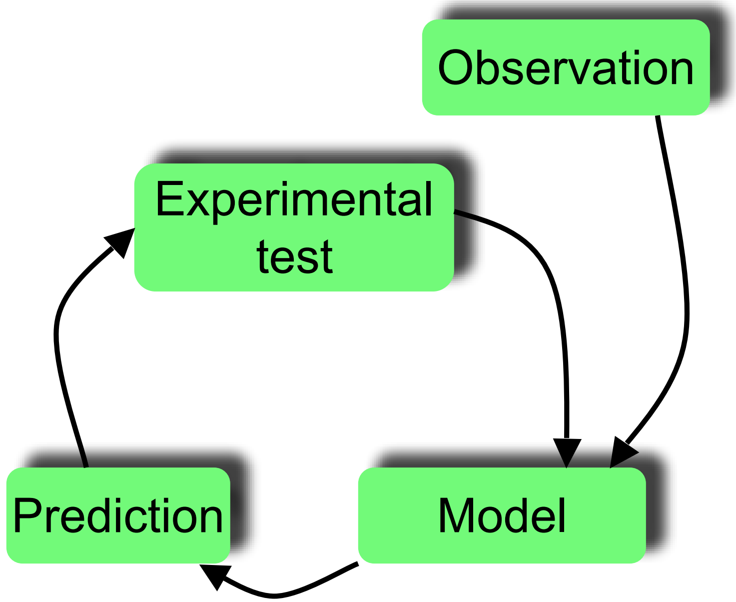
Computer science not at the same level than some other sciences:

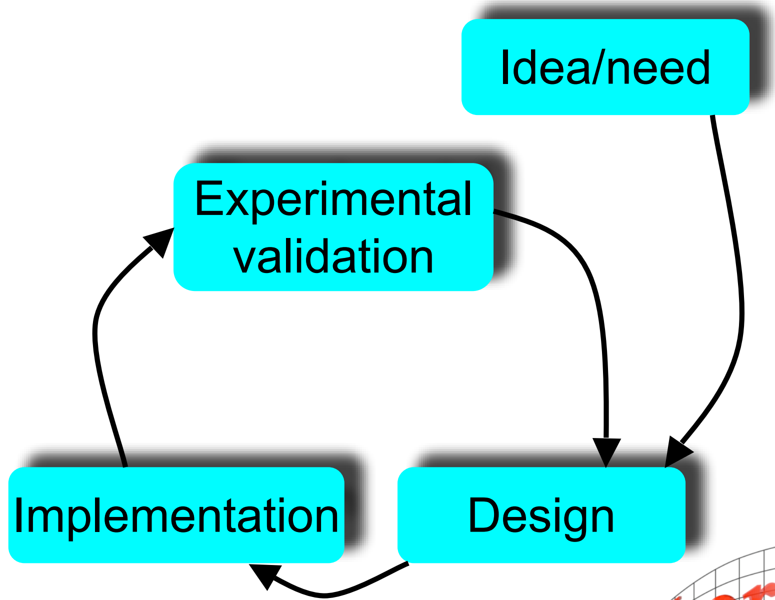
- Nobody redo experiments (no funding).
- Lack of tool and methodologies.

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

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

Micro-Grid

[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).

Wrekavoc a tool for emulating heterogeneity

- 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 :

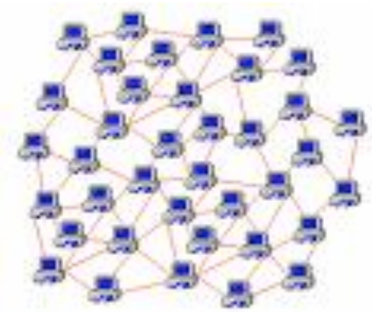
1. Increase the performance (update the hardware)
2. Degrade the performance (by software means)

Solution 2 : Costless and allows for **performance control**.

Tools for configurating the nodes

We want to degrade :

- CPU speed
- Allocatable Memory
- Network bandwidth
- Network lantency



CPU speed degradation

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)

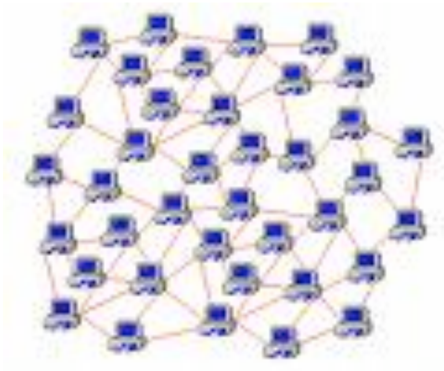
Memory degradation

`mlock` an `munlock`: pins memory pages to physical memory to limit the available memory

Network management

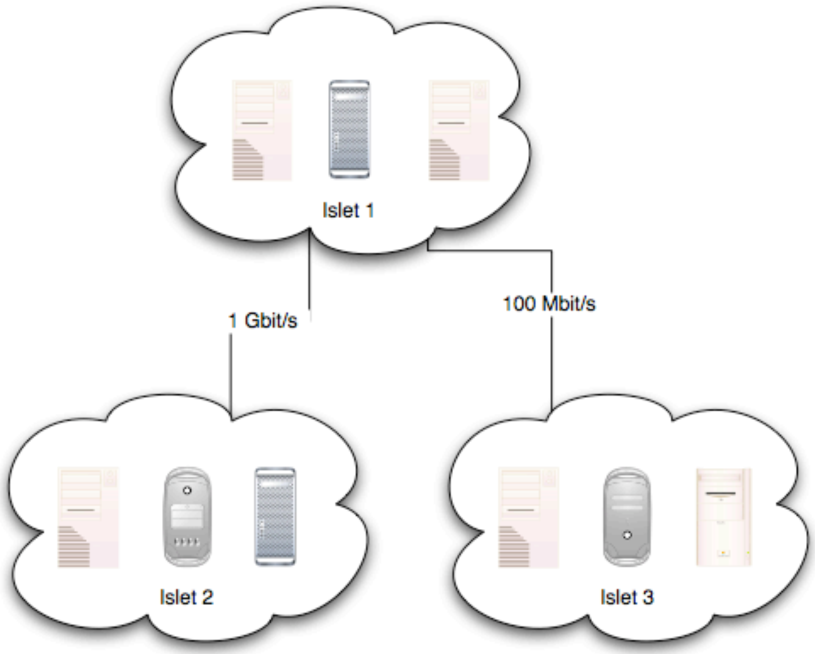
We use (Traffic Control) of iproute2 :

- 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

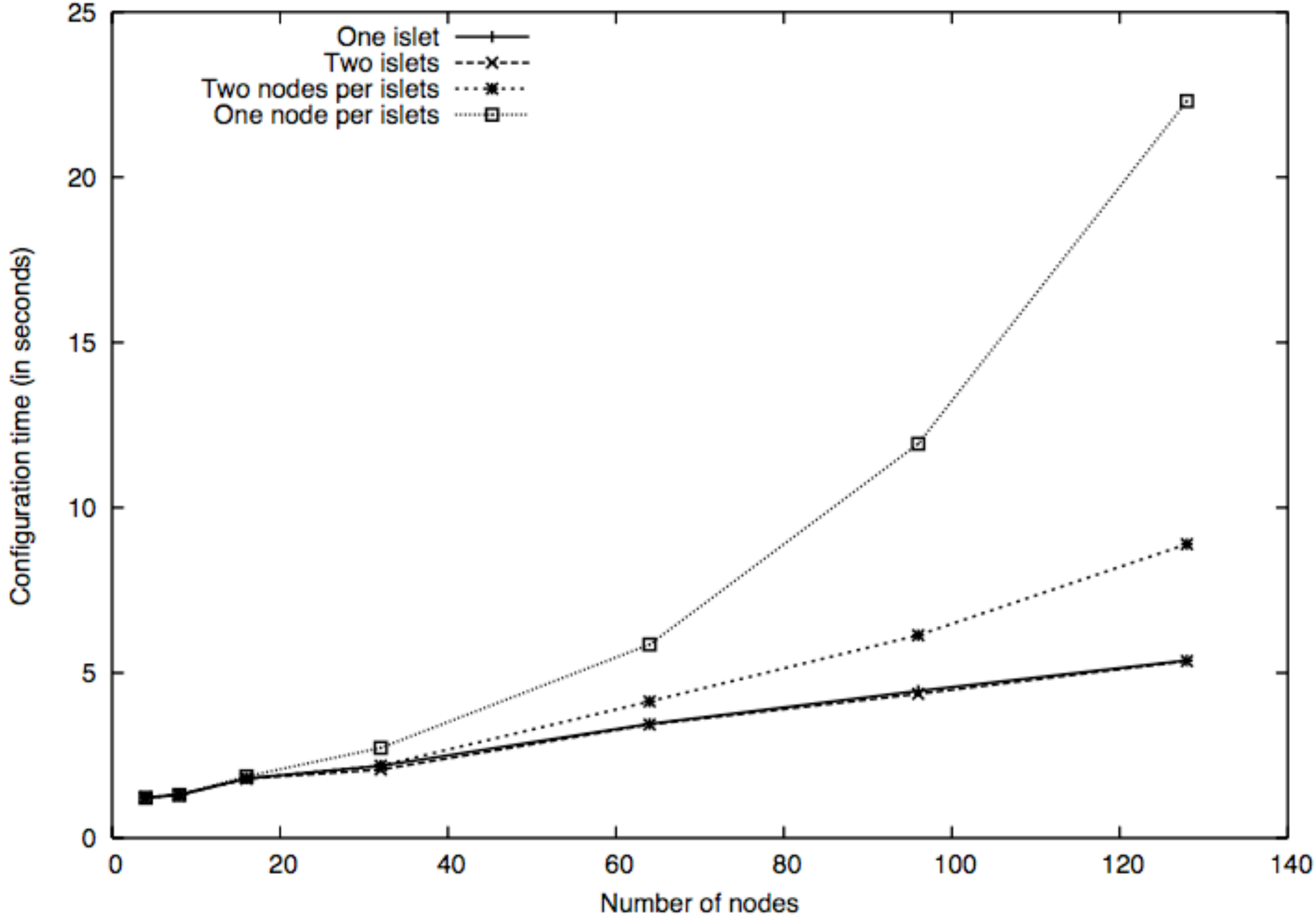
	Simulation	Sandbox/VM	Wrekavoc	Real-scale
Real application	No	Yes	Yes	Yes
Abstraction	Very High	High	Low	No
Execution time	Speed-up	Slow-down	Preserved	Preserved
CPU folding	Mandatory (?)	Possible	No	No
Heterogeneity	Controllable	Yes/No	Controllable	No

Experiments

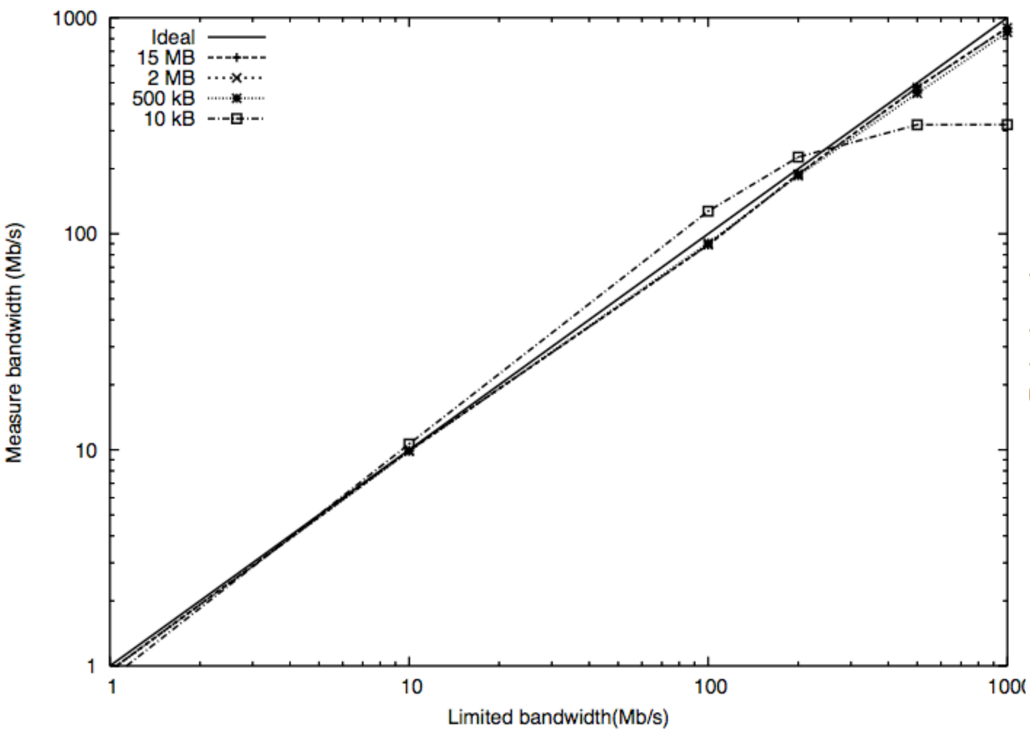
Several experiments on Grid'5000:

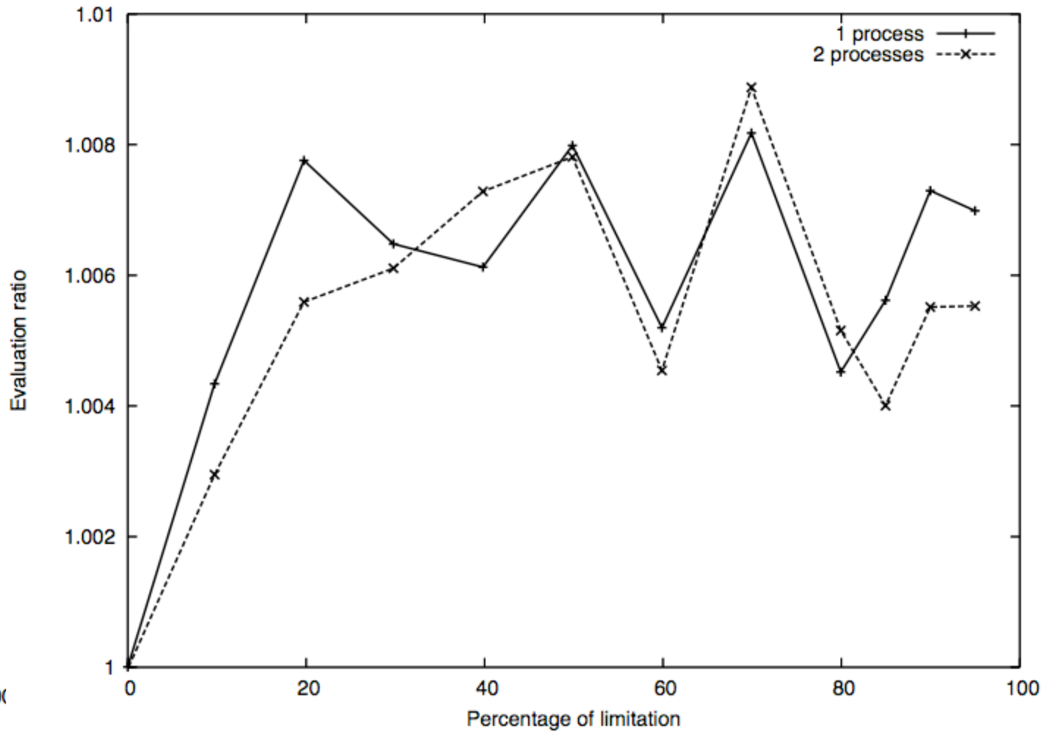
- Configuration time
- Micro-benchmark
- Impact of CPU degradation against available bandwidth
- Realism of Wrekavoc

Configuration time



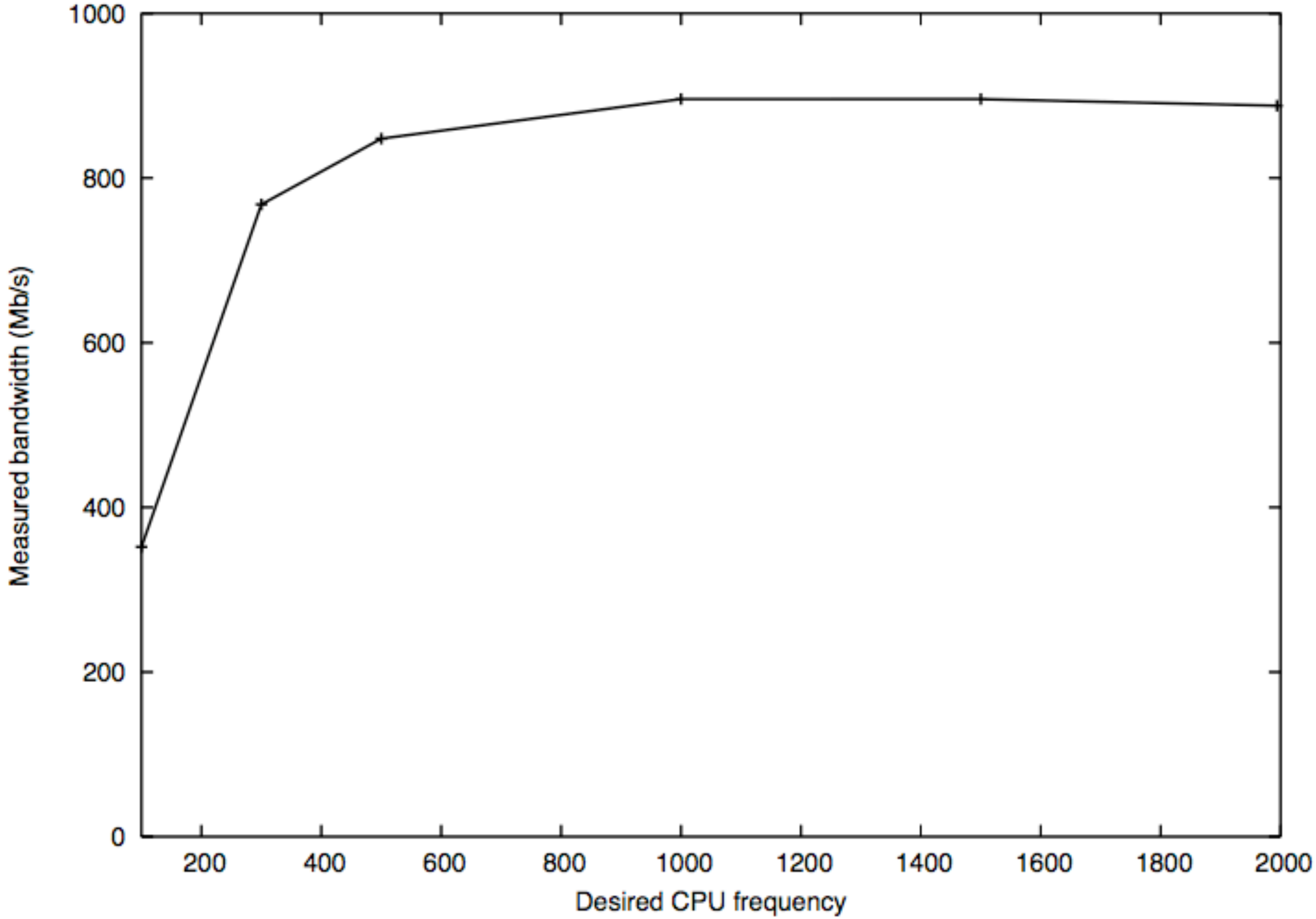
Micro-benchmark





Set latency	1	5	10	50	100	500	1000
RTT	2.12	10.05	20.12	100.06	200.2	1000.05	1999.75

CPU Degradation vs. Bandwidth Degradation

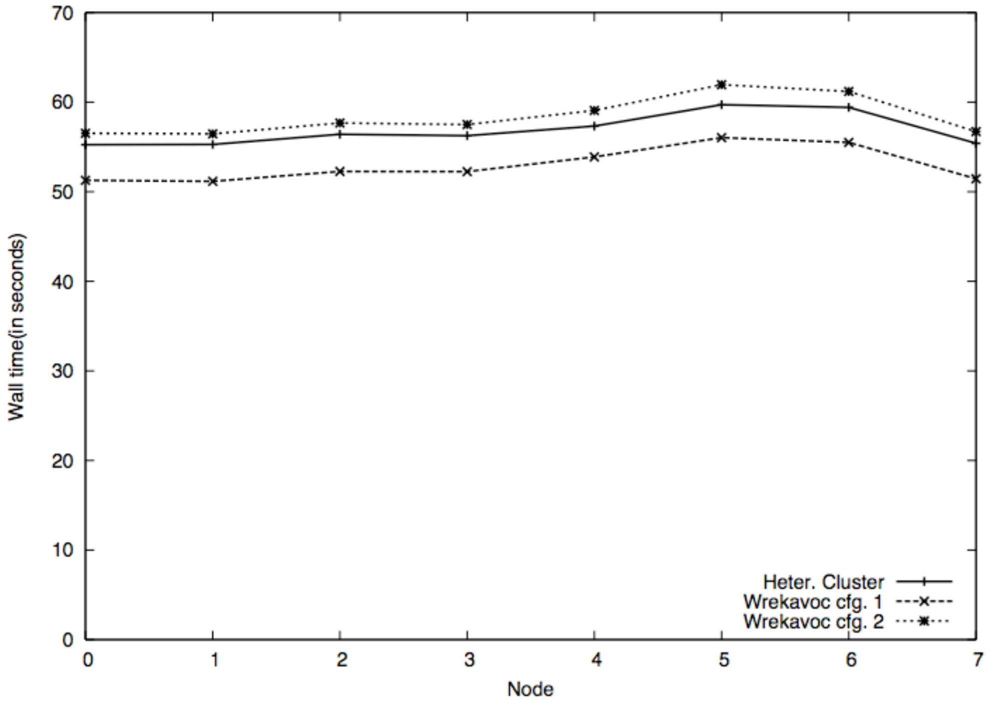


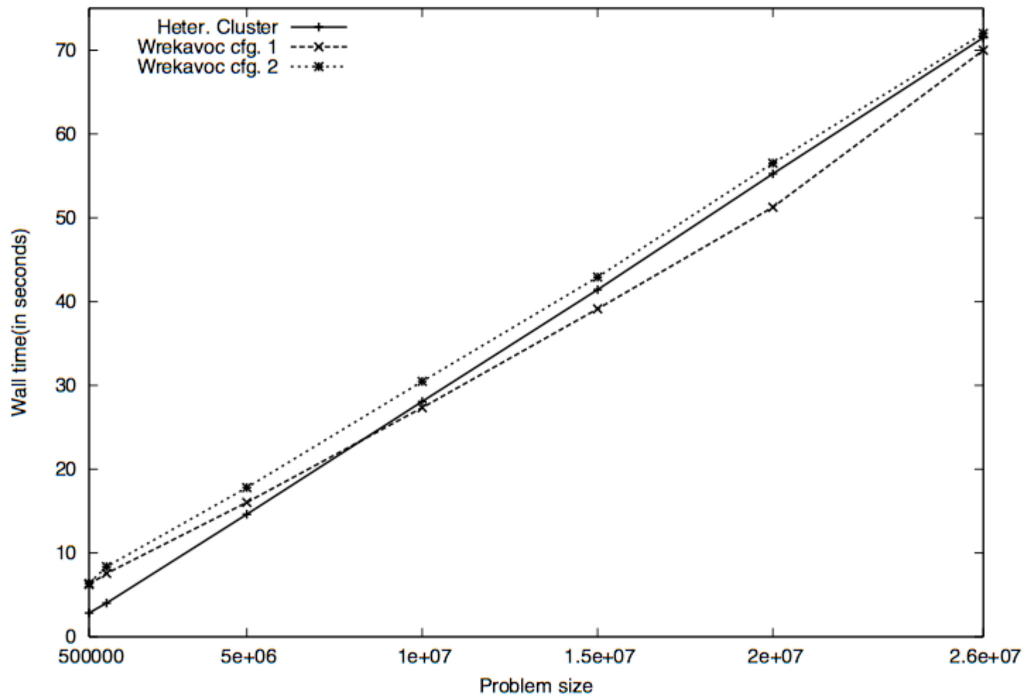
The Realism of Wrekavoc

ID	Proc	RAM (MiB)	System	Freq (MHz)	HDD type	HDD (GiB)	Network card (Mbit/s)	MIPS
1	P. IV	256	Debian 2.6.18-4-686	1695	IDE	20	100	3393
2	P. IV	512	Debian 2.6.18-4-686	2794	IDE	40	1000	5590
3	P. IV	512	Debian 2.6.18-4-686	2794	IDE	40	1000	5590
4	P. III	512	Debian 2.6.18-4-686	864	IDE	12	100	1729
5	P. III	128	Debian 2.6.18-4-686	996	IDE	20	100	1995
6	P. III	1024	Debian 2.6.18-4-686	498	SCSI	8	1000	997
7	P. II	128	Debian 2.6.18-4-686	299	SCSI	4	1000	599
8	P. II	128	Debian 2.6.18-4-686	299	SCSI	4	100	599
9	P. II	128	Debian 2.6.18-4-686	298	SCSI	4	100	596
10	P. II	64	Debian 2.6.18-4-686	398	IDE	20	100	798
11	P. IV	512	Debian 2.6.18-4-686	2593	IDE	40	1000	5191
12	Dual Opteron 240	2048	Debian 2.6.18-4-amd64	1604	IDE	22	1000	3211 and 3207

Description of the heterogeneous environment

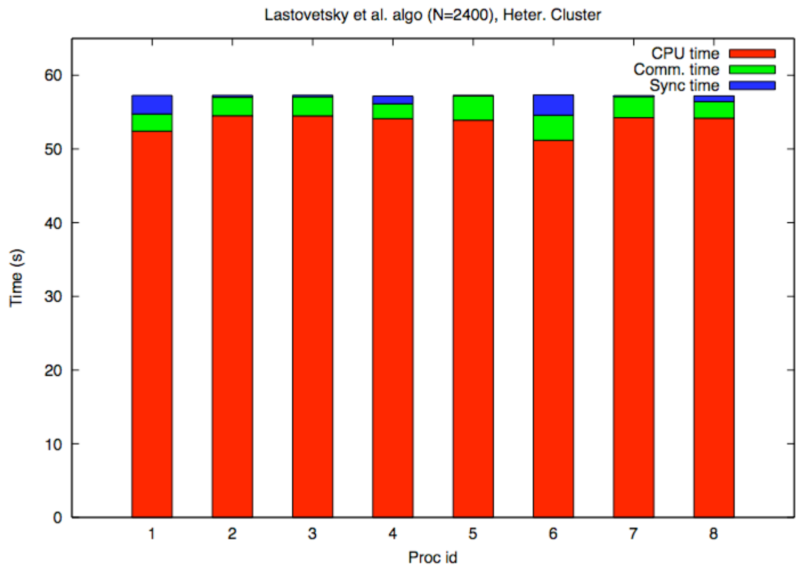
Fine grain computation without load balancing

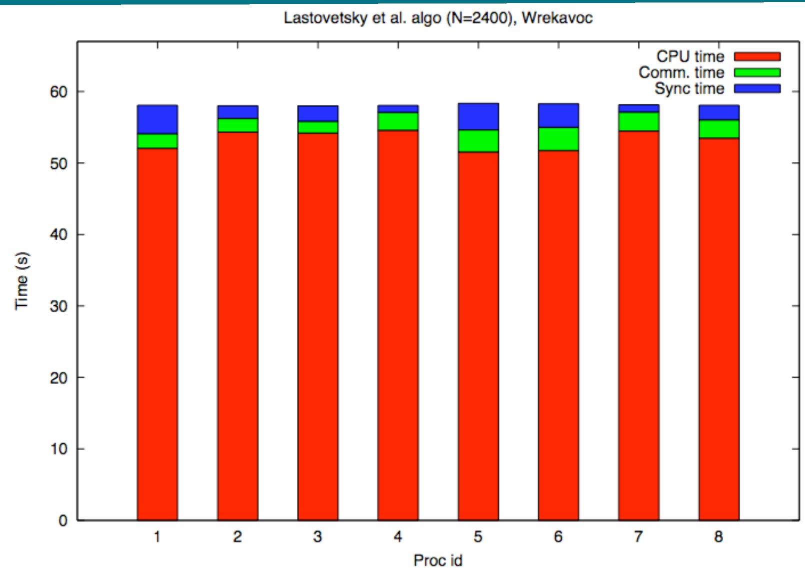


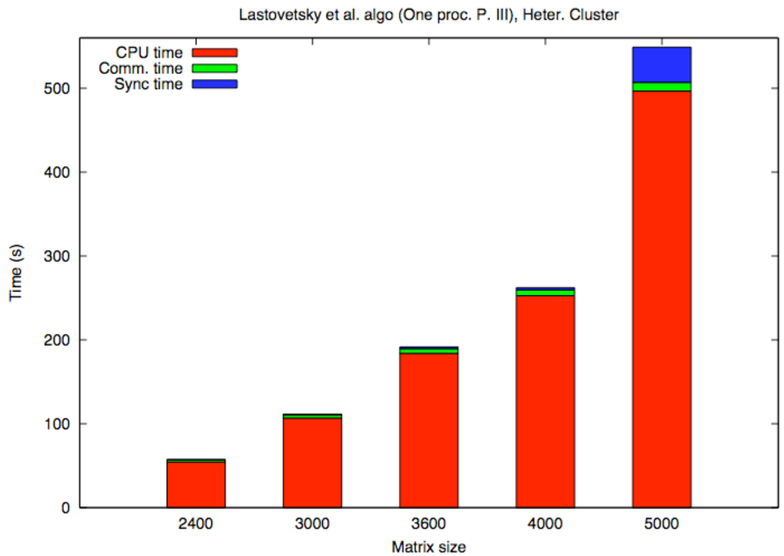


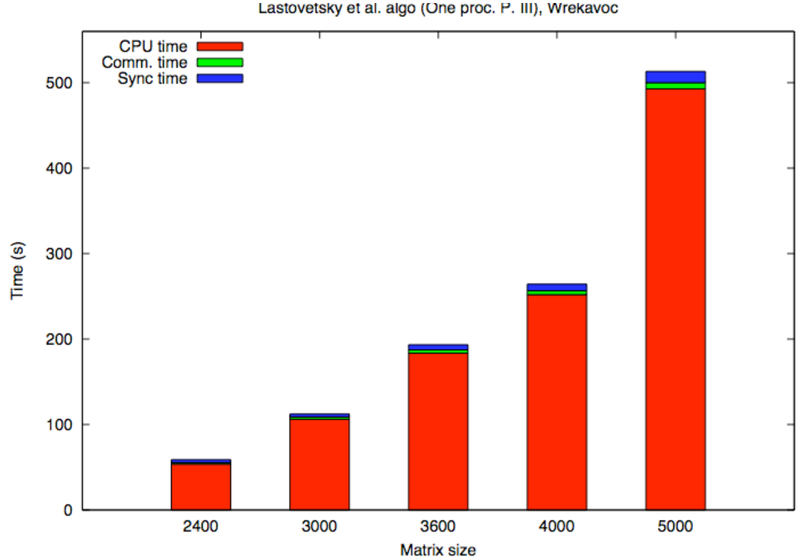
- Gerbessiotis' and Valiant's sample sort algorithm
- Assume heterogeneous env.: Same load on each node

Static load balancing



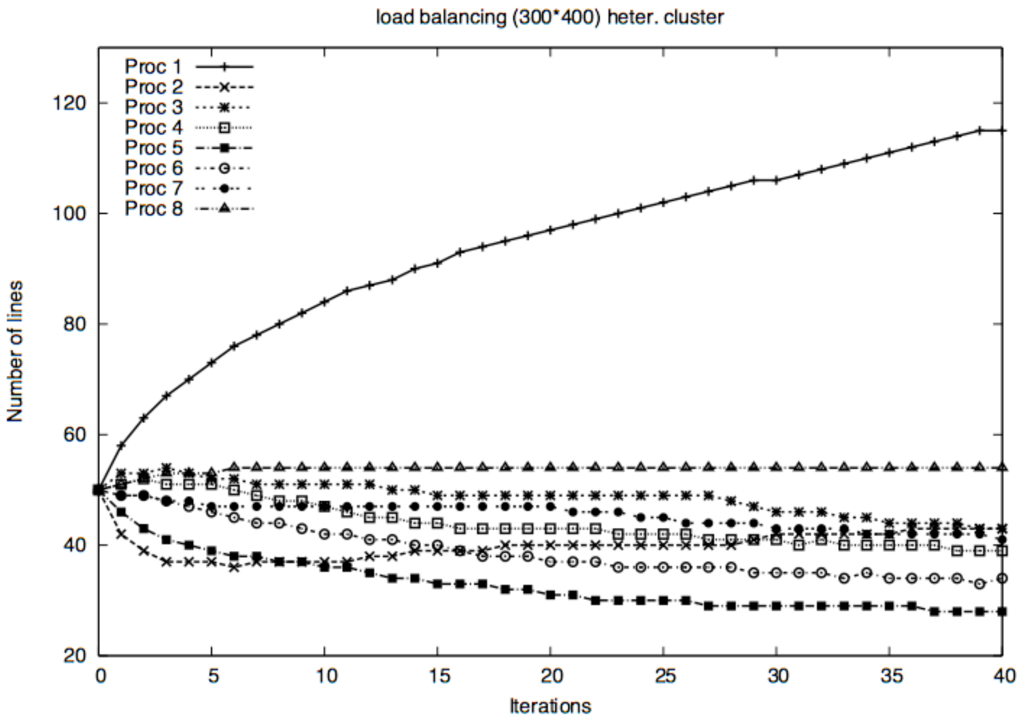


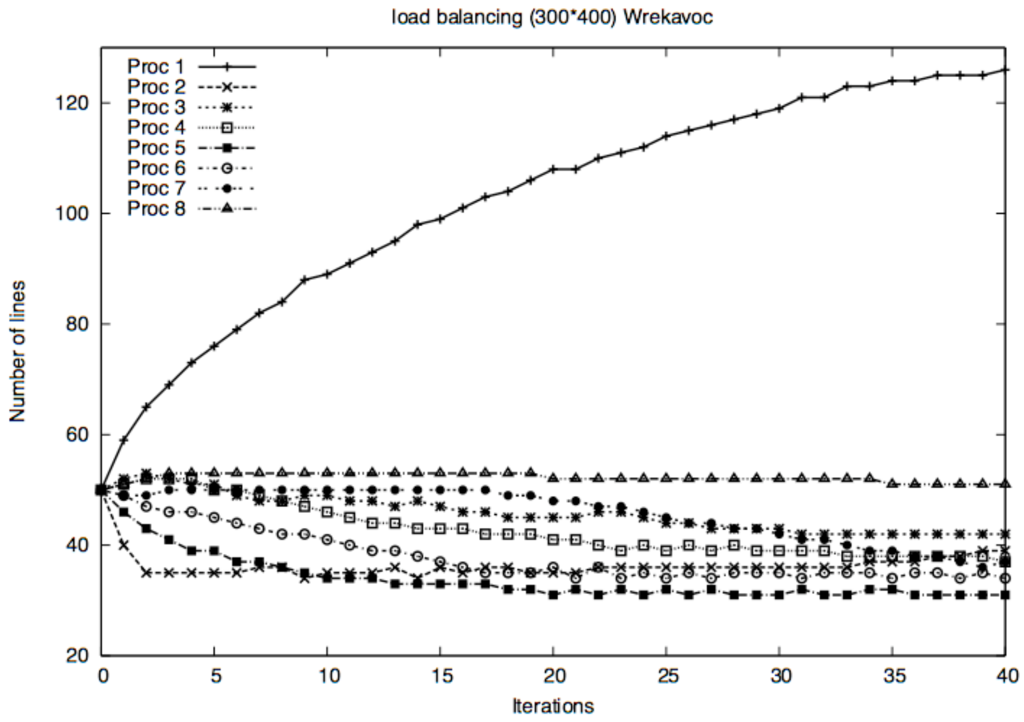




Matrix-multiplcation on heterogenous env. [Lastovetsky et al. 2004]

Dynamic load balancing





Advection diffusion program (kinetic chemistry)

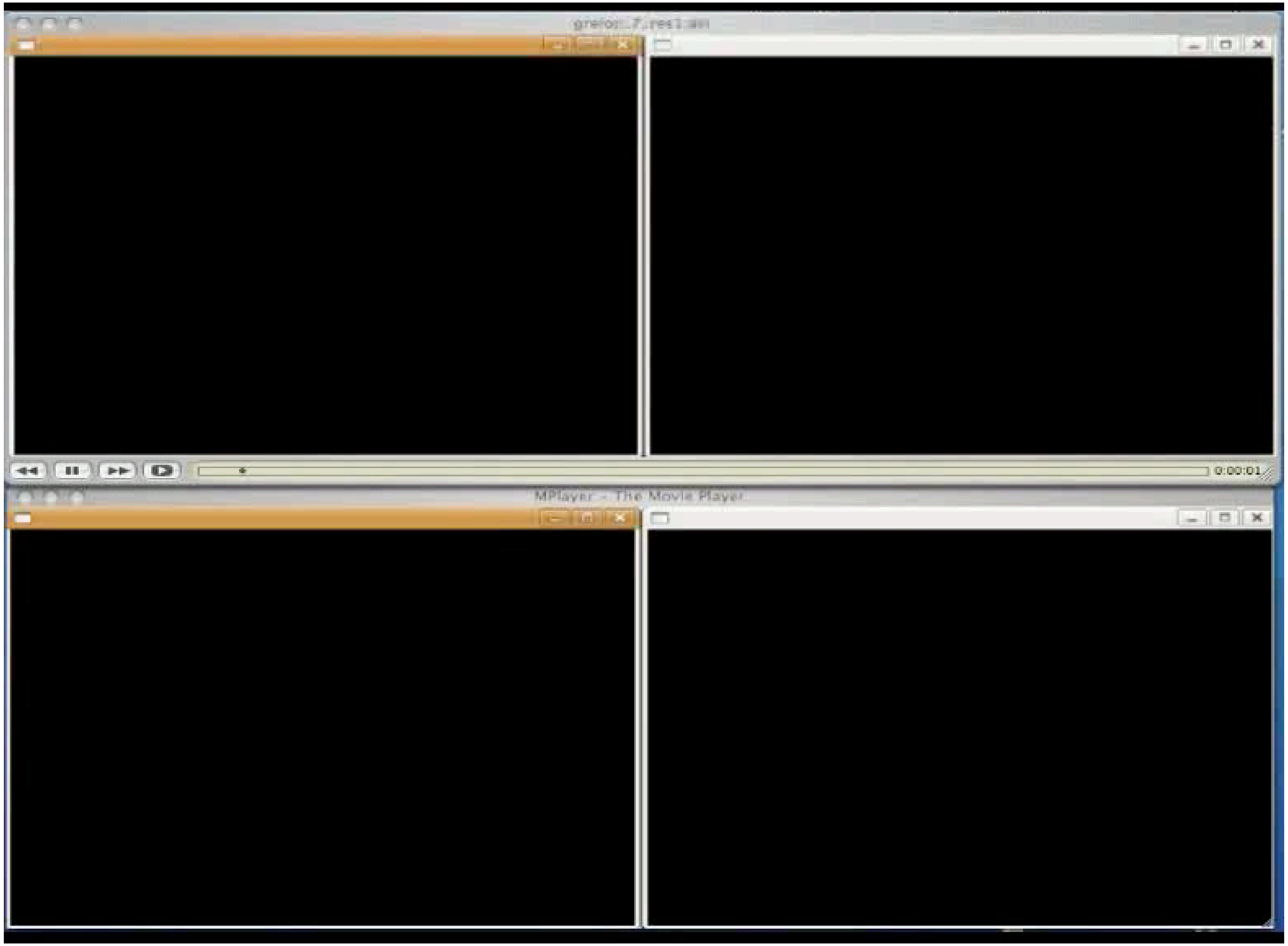
Iterative computation (load exchange at each iteration)

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)

Wrekavoc

Heterogeneous cluster



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/`