Performance Study of Multicore Library on Grid'5000

PaSTeL and Expo

Brice Videau (MESCAL) and Érik Saule (MOAIS)

Supervision : Jean-François Méhaut and Olivier Richard

LIG
CNRS – Centre National de la Recherche Scientifique
Institut National Polytechnique de Grenoble – INPG
INRIA – Institut National de Recherche en Informatique et en Automatique
Grenoble 1 Université Joseph Fourier – Sciences, Technologie, Médecine

June 3rd, 2008

Context and Objectives

Context: Evolution in Processor Architecture

Toward multi-core architectures

- In order to limit thermal output:
  - Decrease in frequency
  - Increase the number of cores to make up for the performance loss
- The trend is going strong
- In a near future every computer will be multi-core

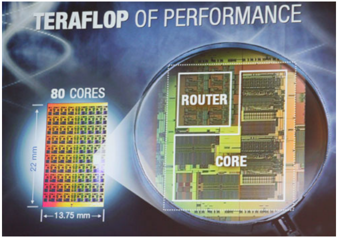


Figure: Intel prototype using 80 cores.

Problematic: What Can Be Expected of Modern Architectures?

As the number of cores increase, the available work per core decrease

Moreover other parameters increase complexity:

- Different hardware
- Different programming paradigms

How small a task can be parallelized on these architectures using these paradigms?

What are the limiting factors?

Objectives: Build an Efficient and Tunable Parallel Library

Two parts: Runtime and Algorithm

Runtime:

- Competitive
- Tunable
- Simple

Algorithm:

- Comparable to others
- Allow the design of a use case

Objectives: In Depth Study of the Parallel Library

Study the performance of the parallel library:

- Variation of internal parameters
- Variation of hardware

Comparison with other parallel libraries.

Problematic - 2: Large Scale experiments

Performance study are long especially when many parameters vary:
$\Rightarrow$ Use Grid'5000 to parallelize experiments.

Grid'5000: cluster of cluster distributed among France

But:

- Many varying parameters
- Many machines involved
- Many results generated

Objectives - 2: Experiment Engine

Build a middleware to conduct experiments on light-weight grids.

We need:

- Fine monitoring of processes: stdout, stderr, return status
- Must not be (too) intrusive
- Fine control of resources: quality, hardware, software
- Native archiving mechanism

Talk Outline

PaSTeL

Overview

Architecture

Algorithms and threads:

- One (or several) main threads
- Execute sequential program(s)
- Work on and gather results of parallel algorithms

Execution model: work-stealing

- *n* threads waiting to steal a main thread
- Have to be very reactive

PaSTeL is modelled and on simple algorithms its performance can be predicted
Reference values are obtained at runtime

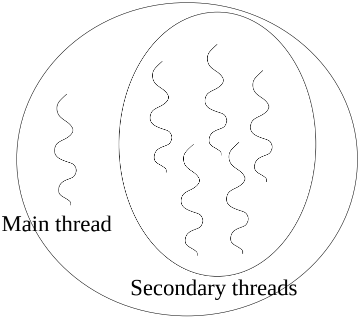


Figure: Threading model

A subset of the STL

STL: Standard Template Library

A collections of algorithms in C++

Parallelized versions exist, thus it is a possible reference.

We Implemented a subset of the STL in PaSTeL

Example

Example use of PaSTeL:

```
#include <pastel_algorithm>
int main (void)
{
  float array[SIZE];
  /* ..... */
  //std::sort(array, array + SIZE);
  pastel::sort(array, array + SIZE);
  /* ..... */
}
```

Methodology

Figures:

- Comparison between STL, MCSTL, TBB and PaSTeL
- Fixed number of threads
- Variable data sizes

Libraries:

- STL: Standard Template Library : sequential
- MCSTL: Multi-Core Standard Template Library
- TBB: Threading Building Blocks

Platforms

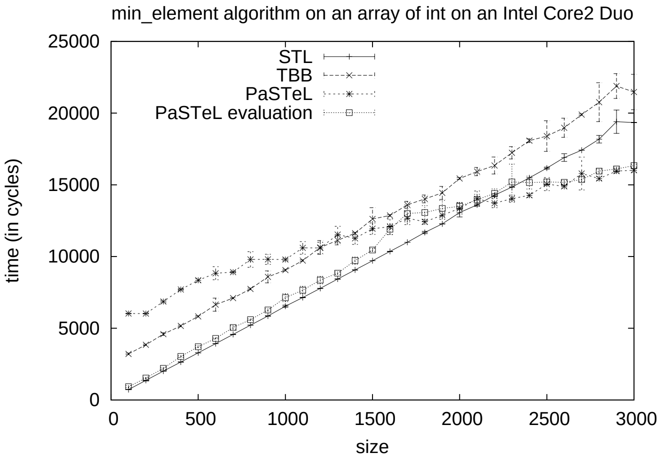
Idkoiff

- 8 * AMD Opteron 875: 16 cores
- Frequency: 2.2GHz
- Cache size: 1024kb per core
- 8 memory banks: 8 * 4 GB : 32 GB
- System: Linux 2.6.23
- Compiler: gcc 4.2.3
- Thread number: 8

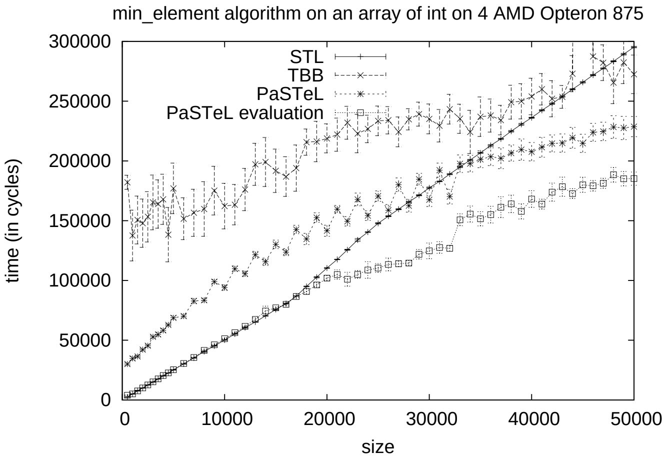
Nedni

- Intel Core 2 Duo: 2 cores
- Frequency: 1.8GHz
- Cache size: 2048kb shared
- Memory: 2GB
- System: Linux 2.6.23
- Compiler: gcc 4.2.3
- Thread number: 2

min_element : nedni



min_element : idkoiff



Expo

Expo: an Experiment Engine

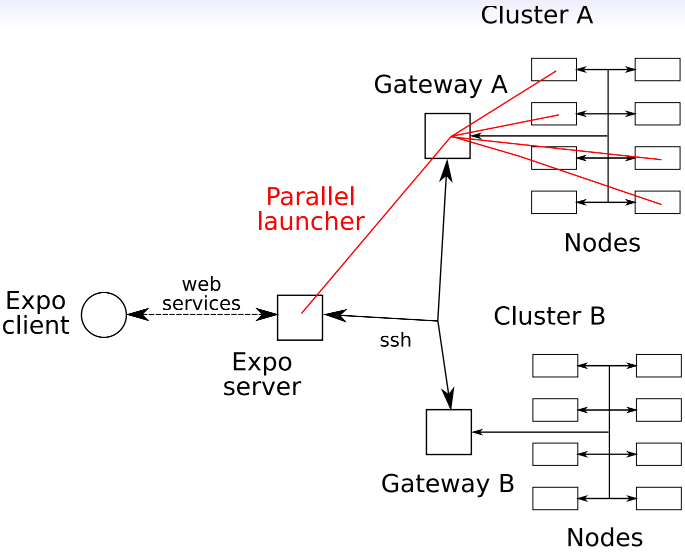
Expo aims at being:

- Tailored for lightweight grids...
- But portable
- Easy to use...
- But powerful

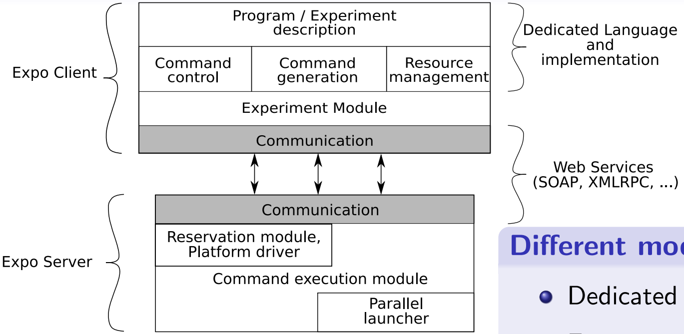
Expo:

- Supervises experiments
- Archives experiments
- Is modular
- Is fault tolerant

Typical Usage



Expo Modular Architecture



Different modules in Expo

- Dedicated Language
- Experiment Module
- Reservation Module
- Command Execution Module
- + Parallel Launcher

Expo Dedicated Language: Overview

Based on Ruby, an interpreted object oriented language. The Expo Language presents different abstractions to the user:

- Resources and Resource Sets
- Tasks and Parallel Tasks
- Synchronous and Asynchronous Tasks
- Reservations
- Parallel Sections and Barriers

Simple Use Case

1. **require** 'expo_g5k'
2. **oargridsub** :res => "gdx:nodes=10,helios:nodes=10"
3. **check** $all
4. **ptask** $all.gateway, $all, "date"
5. id, res = **ptask** $all["gdx"].gateway, $all["gdx"], "sleep 1"
6. res.each { |r| **puts** r.duration }
7. **puts** "moyenne : " + res.mean_duration

Conclusions and Perspectives

Conclusions

PaSTeL:

- Efficient parallelization possible on small datasets
- Several algorithms take advantage of the parallel implementation
- Extremely reproducible results
- Code is still simple enough to be easily modified and modelled

Expo:

- Can already be used on grids with ssh access
- Concise language
- Powerful resource management

Perspectives

Use those two tools to study and model modern architectures

PaSTeL:

- Implement other models than work-stealing
- Use other threading libraries

Expo:

- Improve the DSL with feedback from the new study
- Port Expo to other grid architectures

Annexe