IV Grid Plugtests: composing dedicated tools to run an application efficiently on Grid’5000

Xavier Besseron, Vincent Danjean, Thierry Gautier, Serge Guelton, Guillaume Huard and Frédéric Wagner

xavier.besseron@imag.fr

3rd EGEE User Forum – February 12, 2008

Laboratoire d’Informatique de Grenoble – MOAIS Project
Outline

1. Grid challenges

2. TakTuk, large scale remote executions deployment
   - Scalability
   - Adaptivity

3. Kaapi, large scale adaptive HPC engine
   - Static scheduling
   - Work-stealing scheduling
   - Fault tolerance

4. Conclusions
Outline

1. Grid challenges

2. TakTuk, large scale remote executions deployment
   - Scalability
   - Adaptivity

3. Kaapi, large scale adaptive HPC engine
   - Static scheduling
   - Work-stealing scheduling
   - Fault tolerance

4. Conclusions
Grid challenges

Difficulties for exploiting grids

- Small groups of homogeneous resources (same cluster)
- Some network and CPU disparity among distinct clusters
- Possible traffic isolation in clusters (front node access)
- Resources can fail
- Security of the data
- Certification of the results

Main objective: maintain high performance computation

- Adapt work to processors heterogeneity
- Make use of interconnects hierarchy when possible
- Handle nicely nodes failures
Outline

1. Grid challenges

2. TakTuk, large scale remote executions deployment
   - Scalability
   - Adaptivity

3. Kaapi, large scale adaptive HPC engine
   - Static scheduling
   - Work-stealing scheduling
   - Fault tolerance

4. Conclusions
TakTuk, large scale remote executions deployment

Performs large scale remote executions deployment
- Handle platform topology constraints
- Platform independent
- Insensitive to failing resources

Provides support to applications
- Logical numbering and communication layer
- I/O redirection
- Files transfer
- Can execute distinct commands on distinct nodes
TakTuk scalability

Connects to individual nodes using some standard remote shell command (ssh, rsh, . . . )

- Initiates connections in parallel
- Propagates TakTuk itself to distribute work
- No installation required on remote nodes
Deployment tree can be tailored to Grid topology

- Constrain first connections to front nodes
- Let the engine freely adapt within clusters
**TakTuk adaptivity**

**Local parallelization**
- Uses a sliding window of connection initiation processes
- Adapts the window size to local load (work in progress)

**Work distribution**
- Uses work-stealing to distribute connection tasks
- Sends more work to reactive nodes

**Insensitivity to failing nodes**
- Ignores unresponding nodes (customizable timeout)
- Removes lost connections from the tree
Outline

1. Grid challenges

2. TakTuk, large scale remote executions deployment
   - Scalability
   - Adaptivity

3. Kaapi, large scale adaptive HPC engine
   - Static scheduling
   - Work-stealing scheduling
   - Fault tolerance

4. Conclusions
Kaapi: Athapascan API

Data-flow graph

- Global address space
- Application is described as a dynamic data-flow graph
- Independent of the number of processors

Two C++ keywords

- `Shared<...>`: declares an object in the global memory
- `Fork<...>`: creates a new task that may be executed in concurrency with other tasks
- Access mode is given by the task: Read, Write, Exclusive, Concurrent write

```c++
Shared<Matrix> A;
Shared<double> B;
Fork<Task> () (A,B);
```
Kaapi: Static scheduling (still experimental)

Well-fitted for iterative applications

- Partition the one-iteration graph (SCOTCH, METIS, DSC, ETF, ...)
- Distribute each sub-graph on all the processes
- Repeat the sub-graphs to iterate

Applications

- ANR DISCO [S. Lanteri] Numerical application kernel
- Parallelization of Sofa (simulation of the dynamics of interacting objects)
Static scheduling: 3D-domain decomposition

Preliminary results, Kaapi vs MPICH:

![Bar chart showing mean time for an iteration (s) vs number of nodes for Kaapi and MPICH with 1 and 2 clusters.](chart.png)
Kaapi: Work-stealing scheduling

Well-fitted for series-parallel graphs (recursive applications)

An idle processor steals work to other processors

- Linear speedup if enough parallelism is available:
  \[ T_P = \frac{T_1}{p} + O(T_\infty) \]
- Few steals: \( O(p \times T_\infty) \)
- Provable asymptotic optimality for some algorithms
- Works well with heterogeneous processors

Applications

- ANR CHOC [B. Lecun]: Combinatorial Problem
- ANR SAFESCALE [C. Cerin]: Certification of results
- Grid Plugtest: Solve N-Queens problem
2007 N-Queens Contest

During the GRID@WORK Event, Beijing, China

- Compute the maximal number of solutions to N-Queens
- 6 international teams from China, Poland, France
- Whole Grid’5000 (more than 3800 cores) available for the run

Kaapi/TakTuk Team (6 persons)

- TakTuk integration into ProActive (contest requirement)
- 1 day for parallelizing the application
- 4 days to optimize the sequential parts

Composition of dedicated tools

- ProActive where used to reserve nodes and access clusters
- TakTuk deployed Kaapi processes on all the nodes
- Kaapi processes computed all the solution of the N-Queens problem on Grid’5000 using work-stealing scheduling
### N-Queens Contests results

#### 2006 results: «Prix special du Jury» (not using ProActive)
- N-Queens $N=22$ computed in 8min 22s on 1458 cores
- N-Queens $N=23$ computed in 1h 13min on 1422 cores

Closest competitor: Vrije University (using Satin)
- N-Queens $N=22$ computed in 27min

#### 2007 results: First prize
- Single KAAPI/TakTuk application deployed on 3654 cores
- N-Queens $N=22$ computed in 3min 21s
- N-Queens $N=23$ computed in 35min 07s

Closest competitors:
- ACT (China) deployed 3888 cores
- BUPT (China) computed $N=22$ in 24min 31s on 2925 cores
- Grid-TU (China) computed $N=22$ in 19min 36s on 1735 cores
Fault tolerance

TIC: Theft-Induced Checkpointing

Specialized protocol for work-stealing
- Periodic checkpoints
- Forced checkpoints upon remote steal operation
- [Jafar & Krings & Gautier 2008]

CCK: Coordinated Checkpointing in Kaapi

Specialized protocol for static scheduling
- Coordinates processes to checkpoint
- Recomputes only the required subset of saved tasks to restart
- Provides adaptivity for static-scheduled application
- Implementation and evaluation in progress
Outline

1. Grid challenges

2. TakTuk, large scale remote executions deployment
   - Scalability
   - Adaptivity

3. Kaapi, large scale adaptive HPC engine
   - Static scheduling
   - Work-stealing scheduling
   - Fault tolerance

4. Conclusions
Conclusions

Scalability has been asserted

TakTuk:
- Fast and scalable application deployment
- Fault tolerant
- Provide a communication layer to contact all nodes

Kaapi:
- Use of TakTuk network to communicate with isolated nodes
- Scale up to thousands of heterogeneous cores
- Efficiency is preserved
- But fault tolerance is necessary due to a high failure rate

On-going works

- Static scheduling
- Two fault tolerance protocols: TIC and CCK
- Hierarchical work stealing algorithm for hierarchical networks
Thanks for your attention! Questions?

**TakTuk**

- [http://taktuk.gforge.inria.fr](http://taktuk.gforge.inria.fr)
- Guillaume Huard (guillaume.huard@imag.fr)

**KAAPI**

- [http://kaapi.gforge.inria.fr](http://kaapi.gforge.inria.fr)
- Thierry Gautier (thierry.gautier@imag.fr)
- Xavier Besseron (xavier.besseron@imag.fr)