Adaptability and Dynamicity in Parallel Programming
The MPI case

Nicolas Maillard
Instituto de Informática – UFRGS
21 de Outubro de 2008

http://www.inf.ufrgs.br/~nicolas

The Institute of Informatics, UFRGS
- Federal University of Rio Grande do Sul (UFRGS)
- An international center of excellence in Computer Science and Engineering
  - 88 professors, graduated abroad
    - Brazil (26)
    - United Kingdom (5)
    - France (16)
    - USA (3)
    - Germany (11)
    - Portugal (2)
    - Canada, Belgium, Switzerland...
  - 250+ master & PhD students currently enrolled, 140 PhD already graduated, 1100+ Master already graduated...
  - Undergraduate students exchange with Germany and France
    - Kaiserslautern, TU Berlin, Grenoble, Paris, Strasbourg, Baden-Baden
  - Whatever the metrics, we are part of the “top-5” in Brazil
    - CAPES evaluation, magazine awards, size(s)...

The Parallel & Distributed Computing Group (GPPD)
- 4-5 Professors
  - A. Carissimi, C. Geyer, P. Navaux, N. Maillard, T. Divério
  - 1 postdoc, 12 PhD, 15 Master, plus under-grad. students.
- Working at all the stages of High-Performance Computing
  - Architecture (Navaux)
  - Programming (Maillard)
  - Management (Carissimi)
  - Scientific Computing (Diverio)
  - Grid and P2P Comp. (Geyer)
  - Etc....
Collaborations and Projects

- **Academic:**
  - INPE CPTEC-LAC (São Paulo);
  - UFRJ/COPE (Rio de Janeiro);
  - UFSC, UFRGS, UNISINOS, UNISC, UCS, UCPel, etc... (Rio Grande do Sul)
- **International:**
  - INRIA (France) Équipe Associée since 2005/12
  - Pittsburgh (USA), Berlin
  - Portugal (Lisbon)
- **Industrial:**
  - Microsoft
  - Intel
  - Hewlett-Packard, DELL, etc...

Interaction with Grid5000 (France)

- GSK = a grid of 9 sites, network 1 Gb/s, 4000+ cores today.
  - Plus 2 sites (Luxembourg + Porto Alegre)
- A tool for Computer Scientists.
- Specific middleware:
  - OAR for resource allocation & control.
  - K deploy to deploy whatever image you want.
- Contacts with other grids – DAS (netherlands), NAREGI (Japan)
- The GPPD is about to enter in this Grid.

A few UFRGS clusters

Corrisco

Labtec

Xirú

XD1

Architectures for HPC

- **Network on Chip**
  - How do you implement a hierarchy of memories inside a chip?
    - Henrique Freitas
- **Cache Hierarchy and Multicore Chips**
  - Marco Zanatta
- **Clusters of GPUs**
  - Laercio Pilla

Monitoring parallel applications

- Using 3D representations to visualize the behavior of parallel programs
  - Resources (x, y) vs. Time (z)
  - Rotations, zooms...
- Lucas M. Schnorr’s PhD
  - Double diploma with G. Huard, Grenoble.
A look at the Top-500

• www.top500.org
  – Ranking of the 500 more powerful computers – 31 editions.
  – Dongarra, Meier, Strohmaier.

So how do you use such machines?

• Distributed memory programming
  – Unless you restrict to Multicore architectures.
  – You want to program a cluster of such chips.
• You need support for heterogeneity
  – Larger machines tend to lose homogeneity
  – How about CPUs + GPUs?
• You need support for dynamicity
  – Either because of the platform dynamicity
    • Failures, new resources entering.
  – Either because of the application dynamicity
    • Local refinement of a mesh.
    • Data dependencies
      • Sorting algorithm, for instance.

What do we have?

• MPI, plus:
  – Posix Threads,
  – OpenMP,
  – CUDA.
• MPI-2 provides some basic mechanisms to launch new processes at run-time
  – MPI_Comm_spawn
  – All major distributions have included MPI-2 since 2005.
• In general, the management of the dynamicity is left to the programmer.
  – Implementation dependent.

Other solutions

• Coupling MPI + OpenMP/Pthreads / CUDA
  – It is possible but awkward.
• Adaptive MPI
  – MPI interface on top of Charm++
  – A MPI task = 1 charm task = an object.
  – Facility to migrate a task.
  – Nothing for dynamicity.
• Using another Parallel API
  – Task parallelism: Cilk (MIT), Satin (Amsterdam), Kaapi (Grenoble).
    – “Loop parallelism”: OpenMP, UPC, Fortress (Berkeley)
  – Great... But who is using them?

Typical case of “adaptive MPI program”

• A fluid flows in a square channel.
• There is an obstacle in the middle.
• How does it impact on the velocity/pressure of the fluid?

The adaptativity is provided by the block size of the mesh.
(All the study in SCHEPKE et al., Performance Improvement of the Parallel Lattice Boltzmann Method. SBAC’07)

Performance vs. Block size [Schepke’07]

• The geometry and size of the block impacts on the performance.
  – The 3D blocks win

[Graph showing performance vs. block size]
Using D&C with dynamic creation of processes

- You need to program in order to “enable” the dynamic creation of tasks.
  - Either you enable a dynamic mapping of the iterations (tasks) to the processes/threads
    - See OpenMP
  - Trouble with recursive parallelism
  - either loop parallelism is excluded
    - Use a more “functional” approach, or recursivity.
- Our solution (not original): use Divide & Conquer.
  - Think of Cilk, for instance.
  - Each “divide” phase enables either the creation of parallel tasks, either
    the sequential continuation of the computation.

D&C and MPI-2

- Use MPI_Comm_spawn to (recursively) create new tasks.
  - 1 task == 1 (MPI) process.
- Two problems remain:
  1. How do you communicate with these new processes?
     - The basic mechanism is MPI’s inter-communicator.
     - It only works between parent and children processes.
  2. Where do you schedule these new processes?
     - The basic mechanism is to run the new processes on the same CPU as the parent one.
     - This quickly overloads the original CPU.

D&C with MPI - communication

- First, the communication are limited to parent/child relationship.
- Second, you need a hierarchy of inter-communicators to route the communication.
- Routing the communication this way also enables a distributed scheduling by Work Stealing.

D&C with MPI - scheduling

We have devised two solutions:

- Centralized algorithm [Cera 2006]
  - A process that spawns children first interacts with a daemon.
- Distributed algorithm (Workstealing) [Pezzi 2007]
  - Decouple the notion of task from the notion of process
    - The MPI process becomes a task.
    - Each processor owns a deque of tasks, and a worker process runs an active task.
  - When a processor turns idle the worker process tries to steal tasks from other deques.
    - The routing of the request is done by the tree of manager processes.

A few experimental results

- Centralized algorithm (see [Cera 06])
- Load balancing by Work Stealing (see [Pezzi 97])

Dynamic control of the granularity in MPI

- MPI_Comm_spawn starts tasks as new heavy processes.
- Various MPI implementations enable the mapping of tasks to threads, when you start the initial program.
  - You can even compile a MPI program to get a threaded program.
- What about dynamically creating MPI tasks, whether
  threads or processes, depending on the architecture and of the load of the nodes?

João Lima (MoS)
Dynamic processes + threads

- Mergesort, $3 \times 10^7$ elements – 3 different input sets

Current active work

- It is nice to create tasks... What happens when you have to eliminate some tasks? (Parallelism folding).
  - Get some help from the batch scheduler.
  - M. Cera is currently working on it in Grenoble (batch scheduler OAR).

- Other approach: migrate the MPI processes
  - You do not need to do that in our D&C model...
  - Work started in 2006 with R. Ennes and M. Pillon, currently performed by R. Righi in the “static” case.
  - OpenMPI supports process migration, but not for Spawned processes.

What has been achieved

- Online control of:
  - The scheduling of processes, dynamically created in MPI applications.
    - Requires a “programming style” that enables dynamicity.
    - Solutions both centralized and distributed
    - Limitation: uses online scheduling without any “memory” of the past runs.
  - The granularity of the tasks dynamically created.
    - Threads or heavy processes.
    - Limitation: the max number of threads in a process is fixed at compile time.

More details in our publications

- Ennes, Rafael et al. SBAC’05: “Automatic Data-Flow Graph Generation of MPI Programs”
- Cera, Márcia et al.: WISSPP 2006: “Scheduling Dynamically Spawned Processes in MPI-2”
- Schepke, Claudio et al. SBAC 2007: “Performance Improvement of the Parallel Lattice Boltzmann Method.”
- Righi, Rodrigo et al. SBAC’08: “Controlling Process Reassignment in BSP Applications”
- Lima, João et al. WSCAD’08: “Controle de Granularidade com threads em Programas MPI Dinâmicos” (portuguese)

Fertig!

details, ideas, publis: http://www.inf.ufrgs.br/~nicolas
nicolas@inf.ufrgs.br
http://gppd.inf.ufrgs.br