Parallel Programming (in MPI)

Parallelization Strategies

Dieter an Mey

Center for Computing and Communication
Aachen University of Technology

anmey@rz.rwth-aachen.de

From a serial to a parallel program

Target:

- A large program package takes too much time …
- Want to solve larger problems in future, compute time is likely to explode …

Questions:

- How much effort do you want to put into parallelization?
- Which computing resources are available to you? (or how much money do you want to spend on those resources?)
- Which speed-up do you want to achieve?
- How much data (memory, disk space) do you need to manage?

From a serial to a parallel program

Searching for a concept for a parallel algorithm

- Are suitable parallel libraries available?
- Are there solutions in literature?
- Develop an own concept.

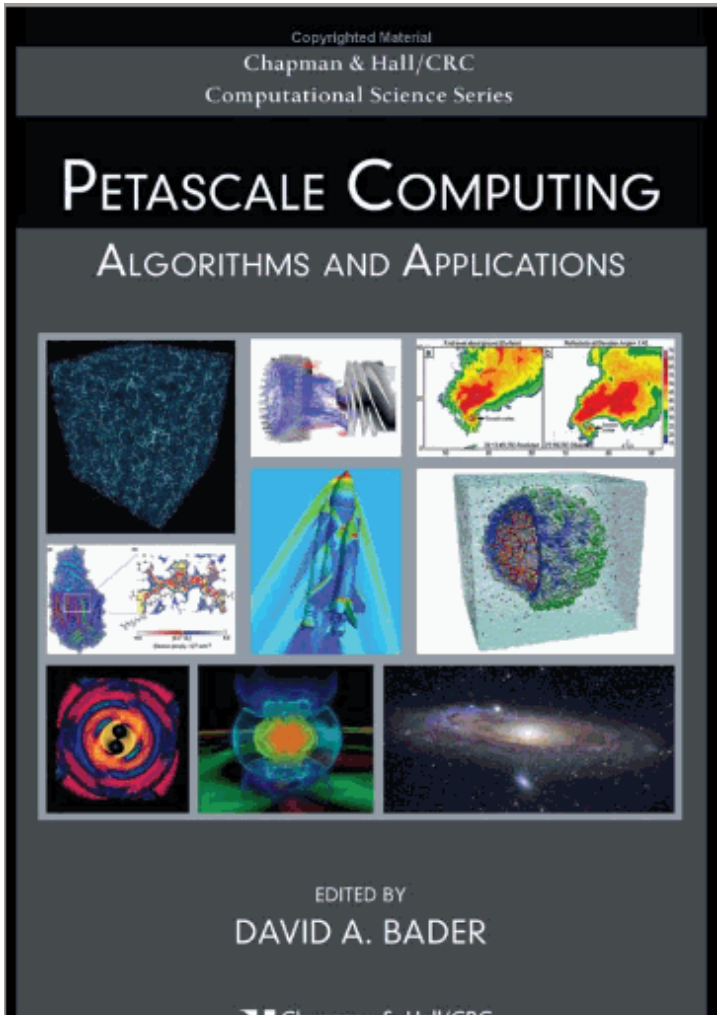
Checking approaches to parallelization.

On which kind of machine architecture does the algorithm fit best?

- Shared Memory Parallelization using OpenMP
- Distributed Memory Parallelization using Message Passing with MPI
- Combine MPI and OpenMP: Hybrid Parallelization
- There are other approaches towards parallelism, which we do not recommend for long term projects.

Are there compute intense small parts in your application which might be offloaded to an accelerator (GPGPU, in future Intel MIC)?

Statistics from the "First Petascale Book"



Keyword	Hits	Remarks
MPI	612	since 1994
OpenMP	150	since 1997 with some 28 hits in our own chapter about OpenMP
threads	109	frequently in the context of OpenMP, 57 in our chapter about OpenMP
C++	87	since 1983
Fortran	69	since 1957
Chapel	49	with some 22 hits in Zima's chapter about Chapel
UPC	30	since 2001
Co-array Fortran	27	since 1998
hybrid MPI/OpenMP	~26	hard to count
C	~20	hard to count
HPF	11	since 1993
X10	9	
Fortress	6	
Java	5	since 1995
Titanium	3	
posix threads	2	1995, Linux since 2003

Petascale Computing: Algorithms and Applications (Chapman & Hall/Crc Comp. Sci. Ser.)
edited by David A. Bader , 2007, 528 pages, 24 contributions, 90 contributors

Know the Technical Limits
Current Limits of the RWTH Compute Cluster

- „small“ nodes: (MPI partition)
  - up to 64 (later 512) nodes with 12 cores,
  - Westmere processors, 3 GHz,
  - 24 or 96 GB Memory
  - Infiniband QDR: <3µs latency, >3 GB/s bandwidth.
- „fat“ nodes (SMP partition):
  - „fewer“ nodes, 128 cores ,
  - Nehalem EX processors, 2 GHz
  - up to 2 TB Memory
  - Infiniband QDR: <3µs latency, >8 GB/s accumulated bandwidth.
- ScaleMP-Cluster („virtual SMP“)
  - Nehalem EX processors with 512 cores, 2 GHz
  - 4 TB accumulated memory
- GPU-Cluster
  - Up to 24 nodes with 2 Westmere processors and 2 NVIDIA GPGPUs

Using MPI …

- Estimate communication costs

- Frequency and volume of necessary data transfers in relation to
- Characteristics of the available parallel computer: compute power and network topology, bandwidth and latency

- Is communication critical in relation to compute time???
  - If not: Ethernet will be sufficient
  - If yes: need a fast network like Infiniband (or shared memory)

- A hybrid approach (OpenMP + MPI) may reduce network traffic and memory consumption.

Typical Challenges when Applying MPI

- In many cases, MPI parallelization requires considerable program changes.
- It opens up a new dimension of program errors !
- Continuing the development of the serial code leads to diverging program versions.
- The parallel program does no meat the expections with respect to speed-up.

Some Recommendations (1)

- Parallelize in **small steps.**
- Try to **avoid errors** rather than have to debug later.
- Take a look at a **toy program** first, which has the same communicaiton structure.
- Carefully test the toy problem using extreme and unlikely test cases.
  - Variation of the processor count 1,2,4,8,... but also 3,7,13,...
  - Variation of the problem size (problem size may be a multiple of the processor count or not, it may be even smaller than the processor count)
- Use a parallel debugger like **TotalView**!
- Use **runtime analysis tools** not only for performance tuning, but also for vizualization of the parallel program behaviour.
- **Finally** implement the well tested communication structure in your application program.

Some Recommendations (2)

- **Try to stick with only 1 program version!**
  - The program may be happy with an MPI dummy library.
  - Use MPI such that the program will be operable with one process only.
  - You may prepare the serial program first, before inserting any MPI commands.
  - Change your program in small steps, try to maintain correctness of intermediate program versions.