Parallel Computing

Hans van Schoot, vanschoot@scm.com

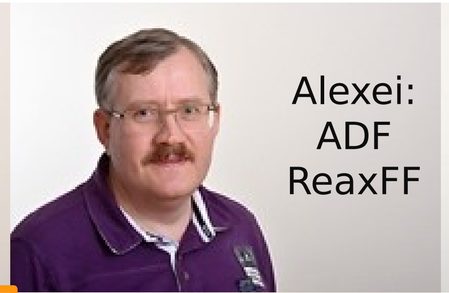
The SCM team

Developers

Business

EU fellows


Olivier:
GUI


Alexei:
ADF
ReaxFF

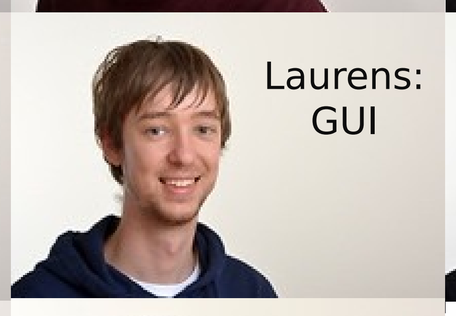

Erik:
ADF
COSMO-RS


Pier:
BAND


Mirko:
ADF
BAND


Hans:
Linux
GPU
Python


Thomas:
DFTB
Scripting


Laurens:
GUI


Ole:
GUI
ReaxFF


Evert Jan:
Adviser

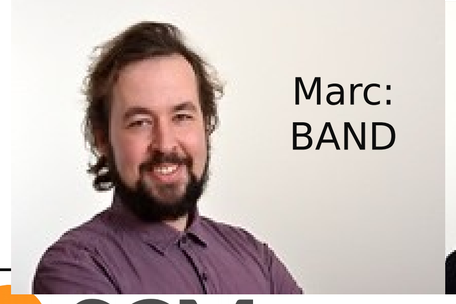

Stan:
CEO


Fedor:
Marketing


Sergio:
Collaborations


Frieda:
Invoices
Licenses


Kitty:
Finance


Marc:
BAND


Damien:
Scripting
MOFs


Anna:
ReaxFF
MOFs


Michal,
Robert:
QM/MM,
excited
states MD

Outline

- What
- Why
- How

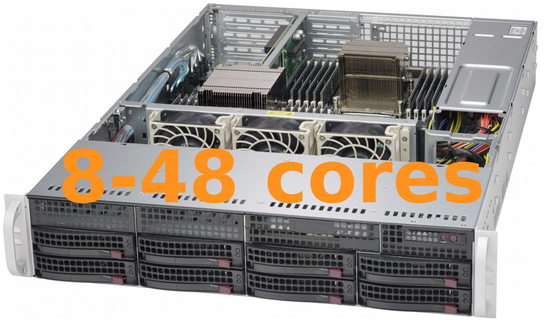
What is parallel computing

Parallel computing = performance

- Parallel computing = using many cores to solve a single problem
  - Faster calculations
  - Bigger systems
  - Better accuracy

- Almost every x86 CPU is parallel (since ~2005)
- Workstations / compute nodes have multiple CPUs
- Clusters / supercomputers have multiple compute nodes

What tools do we have

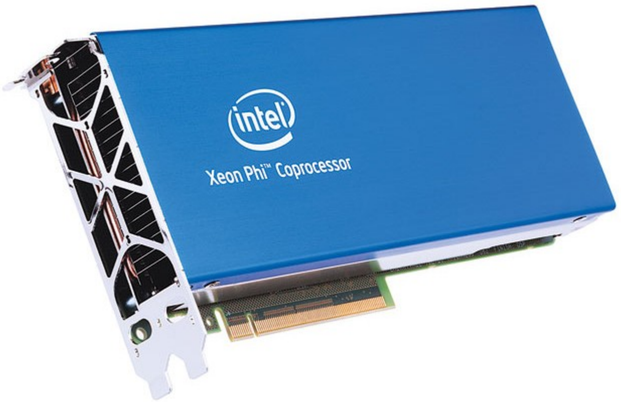


Hardware store equivalent:

What other stuff is out there

- GPU
  - Thousands of cores that have high dependencies and limited resources
  - Excellent raw performance on pre-optimized routines such as DGEMM
  - CUDA / OpenCL language

- Xeon Phi
  - ~60 cores that operate independently
  - Promising for codes that rely less on library routines
  - 2nd gen can be a stand-alone chip
  - OpenMP / MPI / OpenCL



Why we need to go parallel

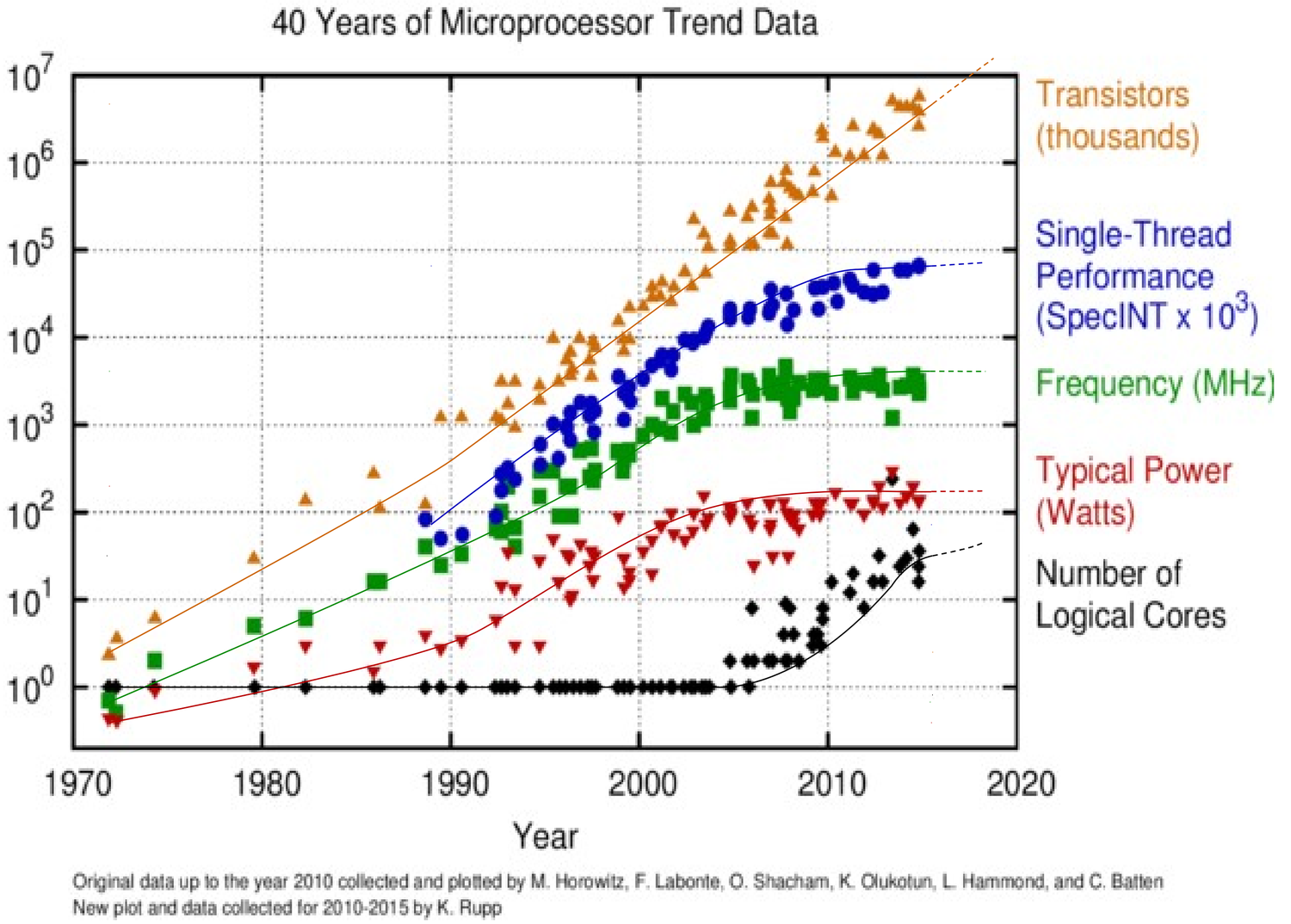
- Performance ≈ frequency * IPC * Ncores

- Increasing “Instructions Per Cycle” is really hard
  - New architecture design + increased transistor count > expensive
- Boosting frequency is easy, but that no longer works :-(
  - Because of power usage! (max ~150W per CPU!)
    $P=cfV^2 + Ps$ (increasing f means that V needs to be higher!)
- Adding more cores at a lower frequency is the solution!
  - if we cut the frequency in half, we can have 4x more cores!
  - Crude explanation: $P \approx f^2$ (lower f needs a lower V)

Why we need to go parallel (NVidia)

“ It’s time to start planning for the end of Moore’s Law, and it’s worth pondering how it will end, not just when.”

Robert Colwell
Director, Microsystems Technology Office, DARPA



How to run ADF in parallel

Laptop / Desktop / Workstation:

- Out-of-the-box parallel calculations
- Windows, OSX and Linux supported

Cluster / Supercomputer (Linux):

- Parallel execution usually works out-of-the-box
- Queue system integration might need some changes in ADFHOME/bin/start
- Read the installation manual for optimizations such as scratch space

- Contact support@scm.com if you need help!

How to run ADF on clusters

- Connect to the machine with SSH
- Set up your calculations with a text editor
- Submit them for execution
- Use adfreport to get results
- You can also use X11 forwarding to run the GUI remotely

OR: Set up your local ADF GUI for remote execution!

- Set up calculation on your local machine
- Select remote machine in ADFJobs queue
- Run your job!

How to run remotely with ADFJobs

- See Installation manual / GUI manual (or the video for Windows)
- Quick demo!

How ADF runs in parallel

- ADF uses MPI, and is available in a couple of flavors
  - IntelMPI for Windows
  - OpenMPI for OSX
  - IntelMPI / OpenMPI / PlatformMPI / CrayMPI / SGI Altix for Linux

- ADF scales from 1 to a couple of 1000 CPU cores
  - Calculation must of course be large enough to run efficiently on many cores

- Do not forget about embarrassing parallel options!
  - 10 parallel jobs with 16 cores per job is faster than 10 sequential jobs with 160 cores!

Parallel Computing Questions?

Hans van Schoot, vanschoot@scm.com