Center of Excellence for Computational Biomolecular Research

The Future of HPC

Exascale and Challenges

Partners

Funding

Horizon 2020
European Union Funding
for Research & Innovation

Reusing this material

BY NC SA

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.

Outline

- Future hardware & architectures
  - Architectures & exascale
  - Processors
  - Accelerators
  - Memory
  - Impacts on performance
- Software challenges
  - Parallelism and scaling
  - New algorithms
  - What about software that does not scale?
- Other impacts and developments relevant to HPC

Future architectures

What will HPC machines look like?

What will future systems look like?

	2016	2020
System Perf.	100 Pflop/s	1 Eflop/s
Memory	1.3 PB	10 PB
Node Perf.	100 Gflop/s	1-10 Tflop/s
Concurrency	O(1000)	O(10000)
Interconnect BW	40 GB/s	200-400 GB/s
Nodes	10,000	O(10000)
I/O	2 TB/s	20 TB/s
MTTI	Several days	O(1 Day)
Power	15 MW	20 MW

Compare to Top500 list: https://top500.org/

Processors

- More Floating-Point compute power per processor
  - Only exploit this power via parallelism
  - More cores combined in some way – “fatter” nodes (also accelerators)
  - Vectorisation (greater vector width) – software needs to use for good performance

- Memory bandwidth very important
  - Top500 #1 HPC machine “Fugaku”: Fujitsu ARM A64FX processor
  - A64FX designed to have high memory bandwidth to feed its cores

Accelerators

- Accelerators (Nvidia & AMD) increasingly widespread in HPC
  - Offer excellent floating point performance per Watt
  - “Fatter” nodes (more computing power)
    - challenge will be to “feed” these fat nodes fast enough – through communications with other nodes, fetching data from memory, …
- Investment in design & fabrication follows the money
  - expect accelerators increasingly optimised for AI applications
    - e.g. double precision less important
  - Scientific computing needs for HPC overlap → leverage hardware
    - c.f. emergence of GPUs for gaming

Memory

- Moving data to/from places where compute happens costs energy & time (performance)

- Memory hierarchy will become more complex:
  - Memory will be packaged with processor
    - Increases power efficiency, speed and bandwidth…
    - …at the cost of smaller memory per core
    - High bandwidth memory on chip with processor and with accelerator
  - Additional persistent storage layer between disk and RAM
    - NVRAM and/or SSD – faster and smaller than disk but slower and larger than RAM
    - Close to compute nodes
    - Enables low latency, high bandwidth to computing elements
  - Still unclear how this will be exposed to users & software developers

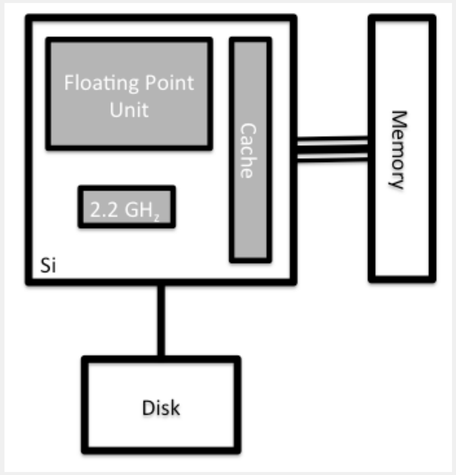
System on a chip

- Instead of separate:
  - Processor
  - Memory
  - Network interface

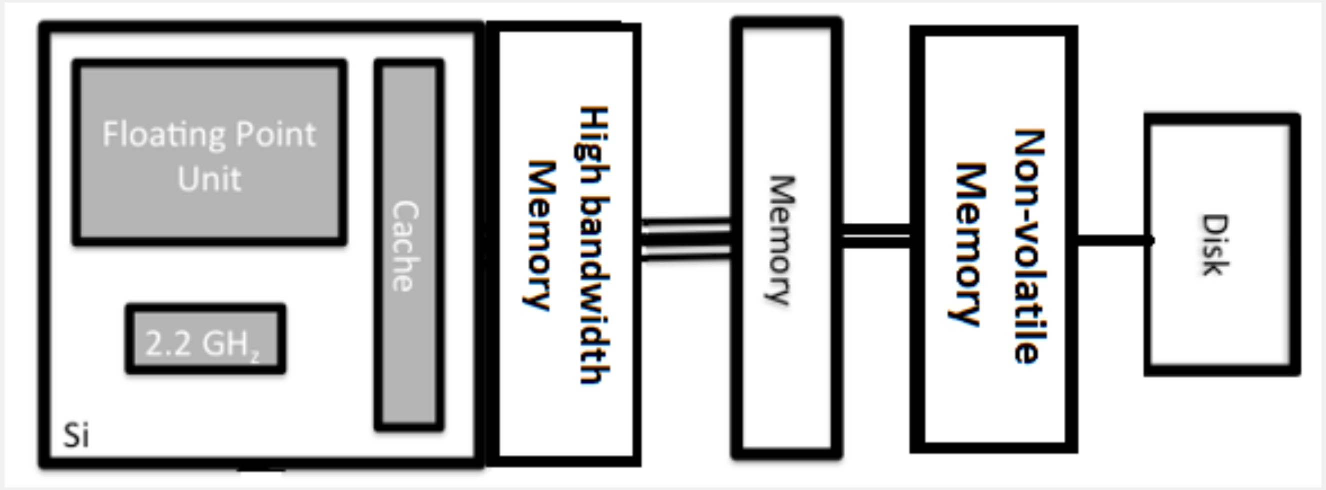
- Will combine these components on a single chip
  - This is what happened to processors in the 70s/80s!
    - Floating point, ALU units used to plug into motherboards directly
  - Reduces latency between components
  - Improves power efficiency (see e.g. Apple Silicon M1)
  - Less scope for customisation
  - If you need more memory than in package you will have to have levels of memory hierarchies

Memory hierarchies

- Moving from:



- To something like:



Software challenges

What does software need to do to exploit future HPC?

What does this mean for applications?

- The future of HPC (as for everyone else):
  - Lots of cores per node (CPU + co-processor)
  - Less memory per core than now
  - Lots of compute power via network interface
  - Increased complexity in memory and IO hierarchy
- Balance of compute to communication power and compute to memory different to now
- Must exploit parallelism at all levels
- Must exploit memory/IO hierarchy efficiently

Algorithms

- New algorithms will be needed to exploit hardware
- Prefer algorithms that run slower on few cores but ultimately contain more scope for parallelisation
- Mixed-precision will become more important
- Try to develop and use numerical libraries that exploit upcoming high-performance energy-efficient accelerators optimised for AI

Applications that do not scale

- If no need for large-scale tightly coupled individual jobs, will still benefit from more of the current size of resources
  - But may be caught out by decrease in memory per core!
  - Options to scale in trivial-parallel way: increase sampling (e.g. ensemble / swarm / replica methods in MD), use more sophisticated statistical techniques
    - This may well be the best route for many simulations

All computing will be (even more) parallel

- All current computers are parallel
  - From supercomputers all the way down to mobile phones
  - Often task-based on 4-8 cores – each application (task) runs on an individual core.

- In the future:
  - More hardware parallelism per device – 10s to 100s cores running at lower clock speeds
  - All applications will have to be parallel
  - Parallel programming skills will be required for all application development.

Cloud Computing (v.s.) HPC

- Cloud computing (AWS, Azure, etc.) has grown in use
- On-demand and flexible
- Not ideal for frequent transfer of very large amounts of data
  - Likely to be a bottleneck
  - On-site computing likely to remain important
- Suitable for high throughput
- Cloud computing historically not had the quality interconnect performance of HPC machines – changing (e.g. Microsoft Azure)

Software Containers & HPC

- Newer HPC machines provide support for software containers (Docker, Singularity)
- Allows more freedom for user customisation of the environment, installed software, etc.
- Facilitate:
  - Management of environments and dependencies (e.g. libraries)
  - Sharing reproducible workflows
  - Portability to different platforms