New Frontier applications and future HPC architectures

Jun Makino
RIKEN Advanced Institute for Computational Science (AICS)

Talk overview

- Past and Present of HPC architectures
- Architectural problems at present and in near future
- “solutions”
- Power wall and “solutions”

Past and Present of HPC architectures

Five eras of evolution of CPUs

I —1969: Before CDC7600
(before fully pipelined multiplier)

II —1989: Before Intel i860 (single-chip CPU with (almost) fully pipelined multiplier)

III —2003:CMOS scaling era (Power $\propto$ size^3)
From i860 to Pentium 4

IV —2022(?):Post-CMOS scaling era (Power $\propto$ size)
From Athlon 64 X2 to Knights Hill(?)

V 2022(?)—: Post-Moore era(miniaturization stops)
???

Evolution of Big Irons

I —1969: Before CDC7600
(before fully pipelined multiplier)

II —1982: Before Cray XMP
(before two multipliers)

III —1992: From Cray XMP to Cray T-90, or Before Fujitsu VPP500 (shared memory vector machines)

IV —2002? From VPP500 to VPP5000
(distributed memory vector machines)

V — now? From Earth Simulator (???)

Evolution of microprocessors

I —1989: Before Intel i860
(before fully pipelined multiplier)

II —2003: From i860 to Pentium 4
(deep pipeline, single core)

III —201X? : From Athlon64 X2 to Xeon Phi
(multicores with SIMD units)

IV — ???

Big Irons and microprocessors

Era		Big Irons	microprocessors
I	multi-cycle MULT	- 1969	- 1989
II	one MULT	- 1982	- 2003
III	multicore	- 1992	- 201X?
IV	distributed	- 2002	???

Observations:

- **Microprocessors follows the evolutionary track of big irons with 20-year delay.**
- **For microprocessors, transition from shared memory to distributed memory should have happened in 2012. It did not.**

Why not distributed memory microprocessors?

Well, what do I mean by “distributed memory microprocessors”? Can there be anything like that?

- If we take the similarity with the Big Irons, it means “VPP500 in a chip”
- This, however, does not make sense, since giving up the cache coherency does not increase the off-chip memory bandwidth.
- With VPP500, by putting one processor to one board, local memory bandwidth is increased.
- With many-core microprocessors, it is not clear how we can increase the local memory bandwidth per core.

Solutions?

Examples of possible solutions

- **True 3D TSV technology (local memory physically on top of each core)**
- **On-chip-only memory architecture**
- **Hope software/algorithm change will solve the problem...**

True 3D TSV technology

- Current 2.5D technology (HBM, HMC or whatever else) is not the solution, since they would give a few TB/s to chips with a few tens of TF.
- We need a single DRAM chip (or a stack of DRAM chips) which can put on top of a processor chip to provide multiple TB/s.
- Technically not impossible, but is the market large enough? (probably no)

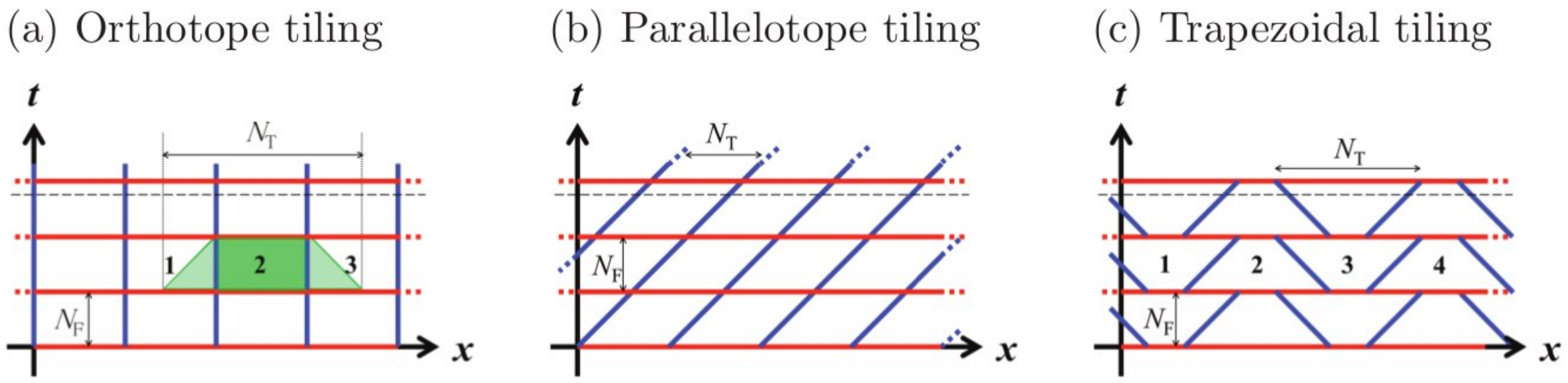
On-chip-only memory architecture

- **If applications can live with on-chip memory only, distributed-memory single-chip many-core architecture makes sense**
- **With 10nm technology, 256-512MB/chip is within reach.**
- **If combined with very low-latency interprocessor communication, it could be used to solve real-world problems very fast.**

Software/Algorithm change

- Most likely to happen (or nothing else is likely to happen)
- HPL runs happily on tiny B/F anyway.
- Particle-based simulation codes do not require much memory bandwidth.
- Stencil calculation? Let's hope temporal blocking will solve all problems.
- Unstructured mesh? Well... How about moving to meshless methods?

Temporal blocking



- Read in a small localized region to on-chip memory (cache), and update that region for multiple time steps.
- Can reduce the required bandwidth to the main memory.
- Fully parallelizable 3D algorithms exist

Summary for Architectural problems

- Current microprocessors are facing the same problems as those shared-memory vector machines faced in early '90s.
- Shared-memory vector machines were superseded by distributed-memory machines.
- It is not clear what will supersede current cache-coherent manycores.
- Most likely nothing will happen. Applications will adapt to tiny B/F.

I hope I will be proven wrong.

How about power consumption

- power $\propto$ # of transistors
- therefore reduction of transistor count per operation means reduction of power
- low-voltage operation (NTV etc) would give another order of magnitude improvement

Not all processors are created equal

How many ~~Intel engineers~~ transistors it takes to do one floating-point op?

GRAPE-3	1991	4K	apl-specific pipeline
GRAPE-6	1997	30K	apl-specific pipeline
GRAPE-DR	2006	400K	SIMD 512-core chip
Cray-1	1976	400K	early vector
Intel i860	1989	600K	Beginning of era III
Earth Simulator	2002	4M	matured vector
NV Fermi	2010	3M	GPGPU
Sandy Bridge	2011	40M	Deep in era IV

Difference of 3-4 orders of magnitude

Why such a huge difference?

- Two orders of magnitude for programmable cores with double-precision arithmetic
- Two orders of magnitude from number format, specialized pipeline, etc

Possible sources of difference in programmable cores

- Deep pipeline
- Register file
- Cache
- Instruction fetch/decode/....
- Memory interface

So how many transistors do you really need to do one multiplication?

Mantissa	# trs.
53	$\sim$ 100K
23	$\sim$ 20K
16	$\sim$ 10K
8	$\sim$ 3K

Quite a large room for "improvement"

Speculations on power consumption

- Reduction by 1-2 orders of magnitude might be possible by streamlining the processor architecture.
- Another 1-2 orders of magnitude by optimizing the word length (need to develop new algorithms).
- Yet another 1 order of magnitude by going to low voltage.
- In total, 3-5 orders of magnitude = 15-25 years.
- Probably more than enough to cover the time to my retirement and beyond. Nothing to worry about.

Summary

- For future HPC architecture, there are two problems: Limit of manycores and limit of power.
- The former will most likely be “solved” by the effort in the application side, though better approaches exist.
- the latter can be ... well, postponed for another 20 years or so.