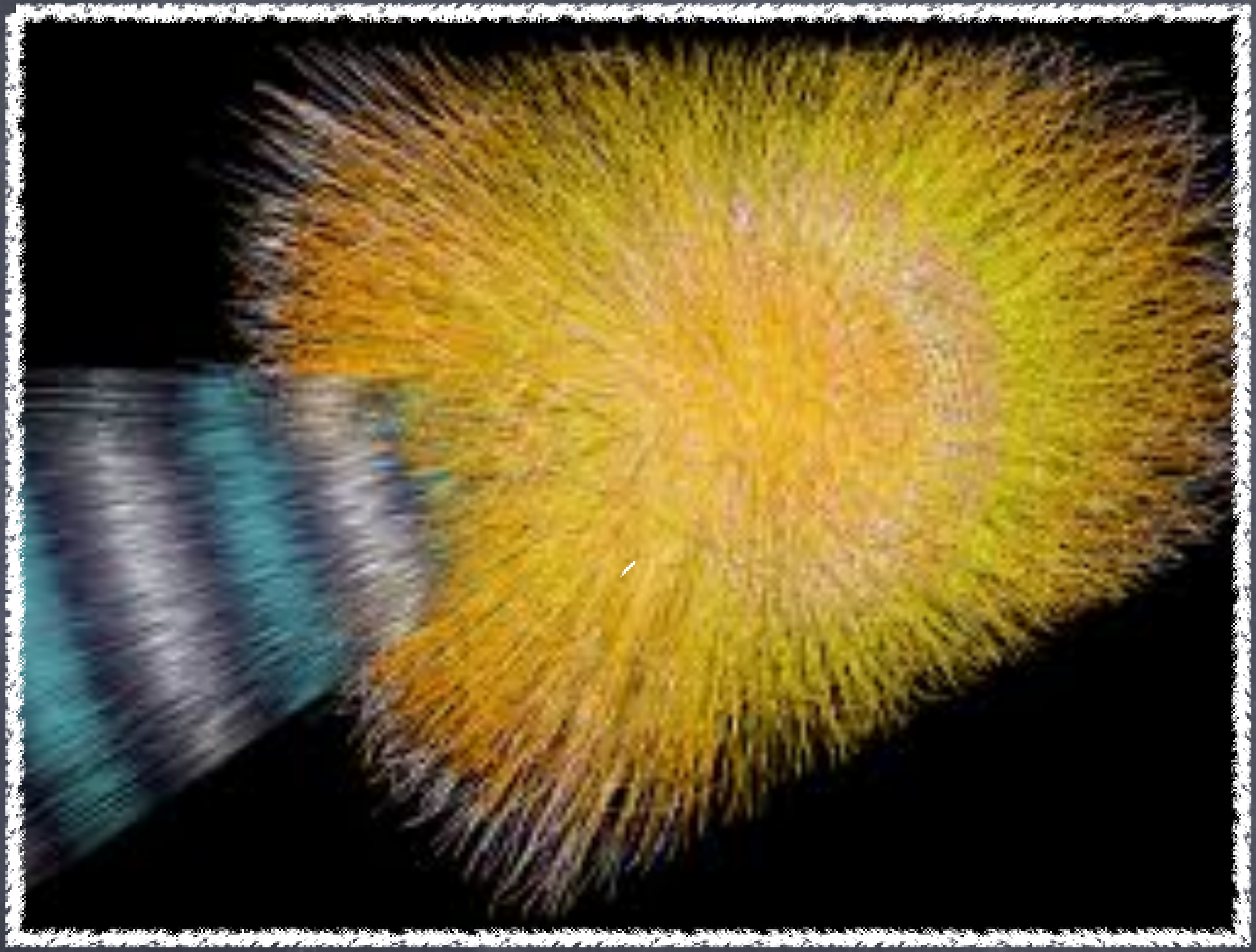
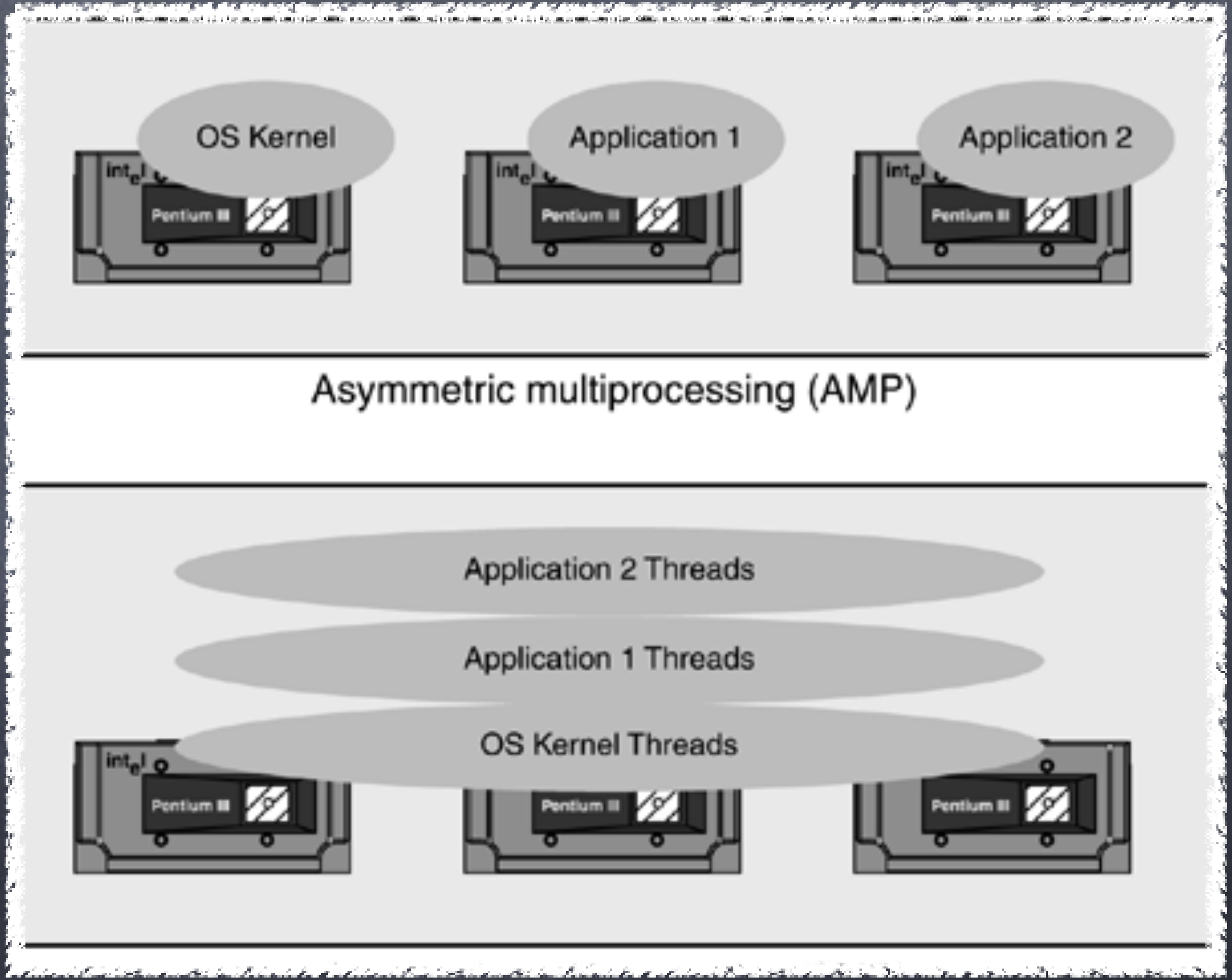


Assymmetric SMP for higher performance

Christoph Lameter, Scalability track 2012

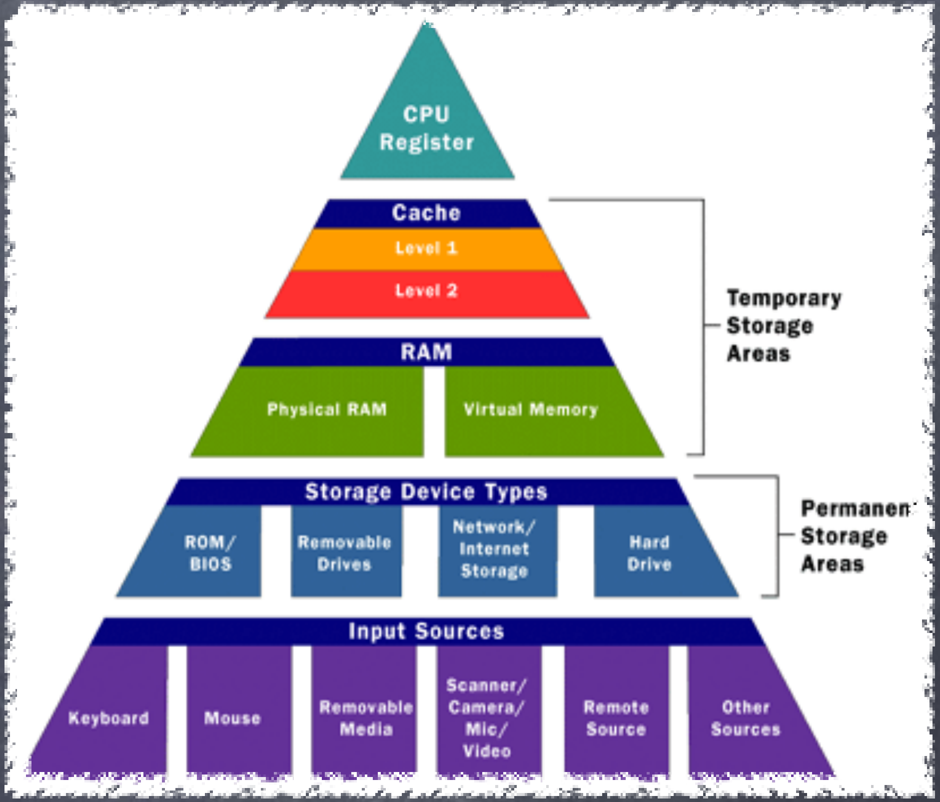
Simple scenario for ASMP

- Three tasks that are shared on 3 processors increase cache footprint, locking overhead etc etc
- Performance decreases



Cpu caches

- L1 32K data/32k code (4 cycles = 1.3ns)
- L2 256k code + data (12 cycles = 4 ns)
- L3 8M(4M). 24 cycles = 8ns
- Local memory = 60ns.
- Remote memory = 100ns.



ASMP advantages

- Single threaded execution possible. Back to the days of the single processor for limited pieces of code.
- Less complex logic
- Easier to maintain
- Full exploitation of hardware speed requires full exploitation of processor caches and reducing cacheline bouncing. Some large piece of code must be running in a hardware context for a reasonable amount of time.

Problems

- Tasks must be doable with the resources available in one hardware context
- Unbalanced execution
- Counter to current scheduler design.

How to do it in some fashion now

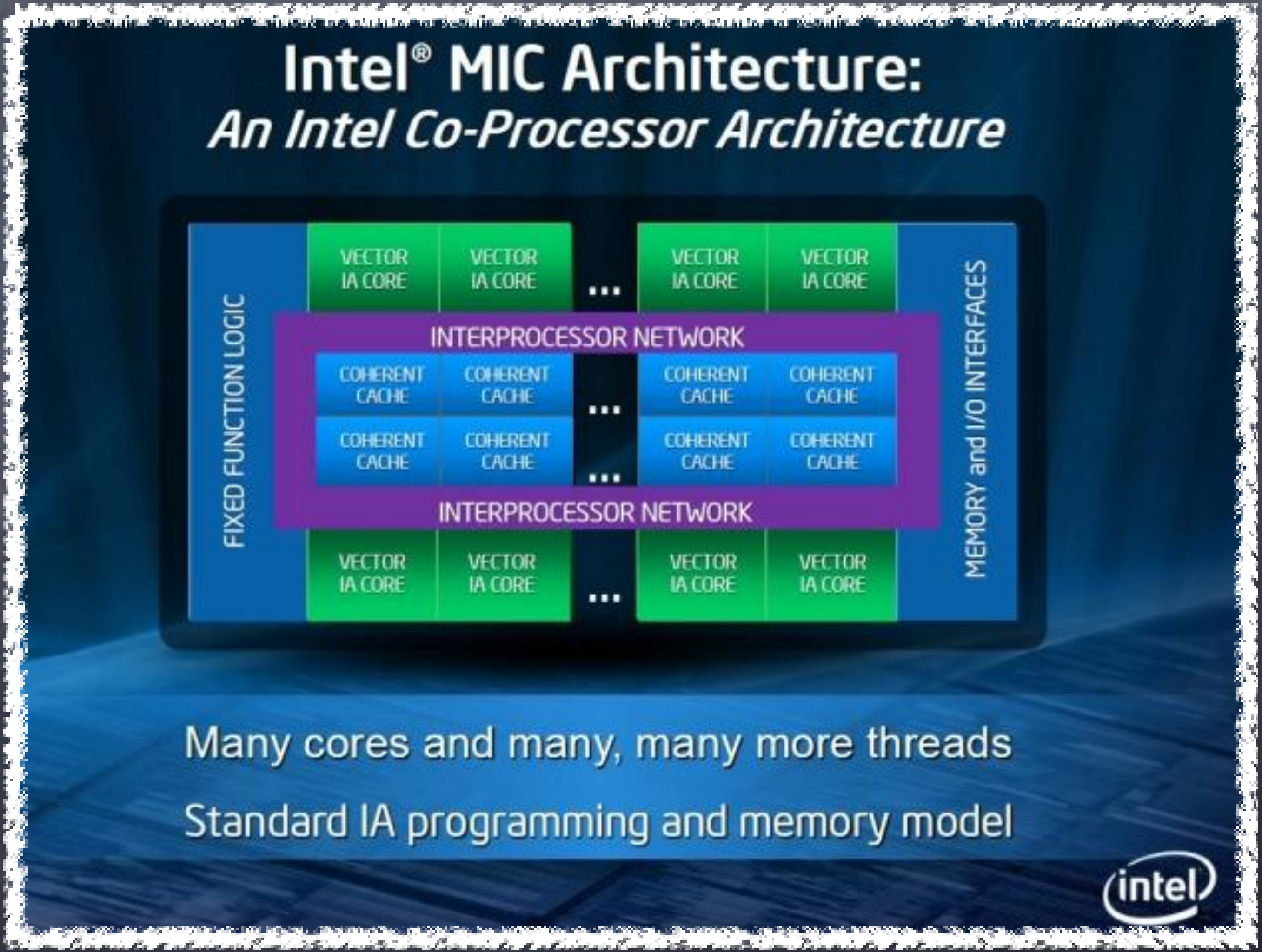
- Restrict init to one processor on bootup
- Isolate cpus to limit scheduler.
- You can pin tasks in user space and set real time priorities to avoid OS interference.

Potential uses of ASMP in the Kernel

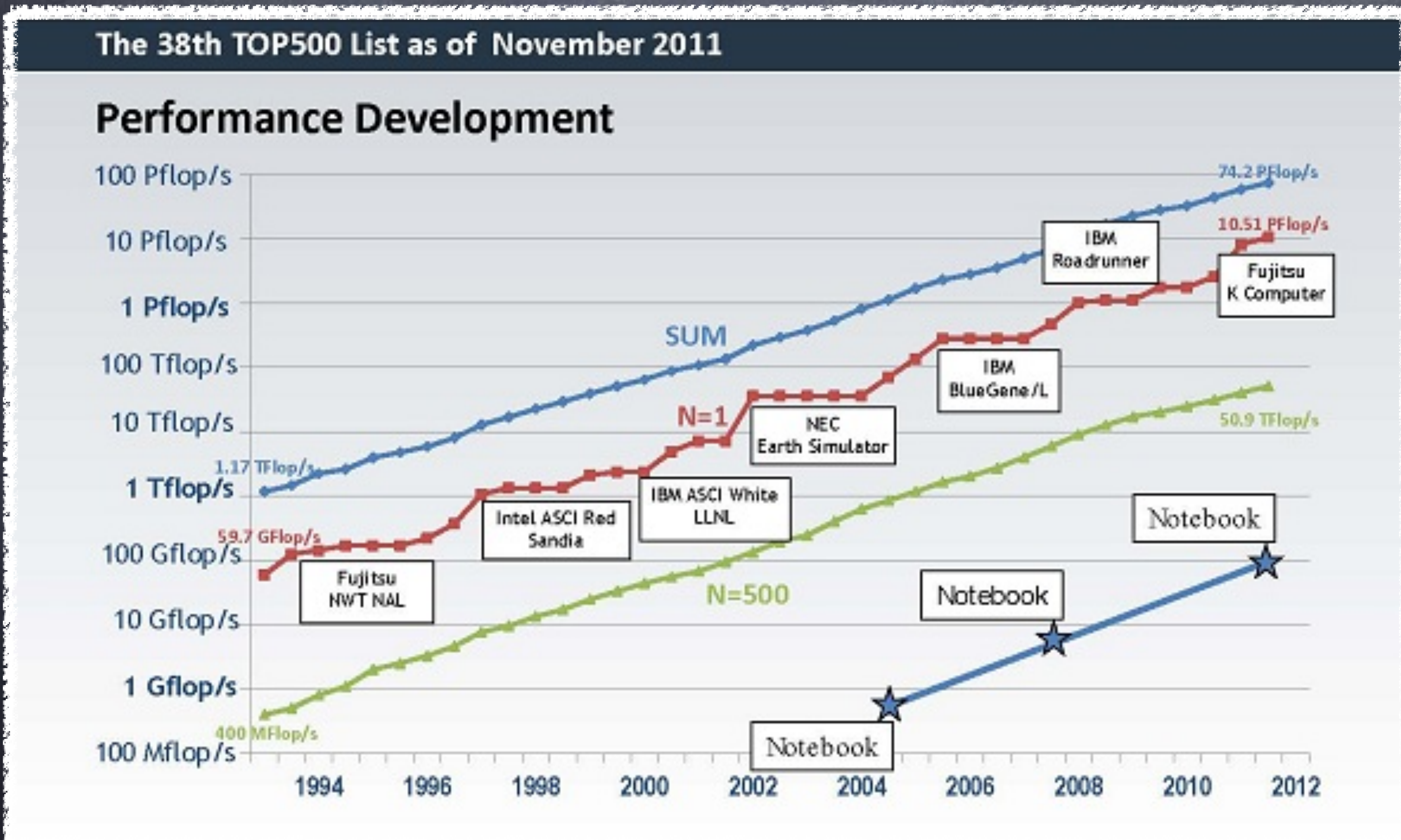
- Reclaim/Swap/Writeback
- Compaction/Migration/KSM
- Exploitation of cpus with different hardware characteristics (low power processing cores on ARM?)
- RCU
- Time processing
- Deferred tasks
- Subsystem specific repeated actions (networking f.e.)

Performance issues with many cores

- 60 cores using x86 processors
- Its .... ummm... too few.. Nvidia has hundred to thousands on GPUs.
- GPU technology shows the way ahead.



Performance evolution

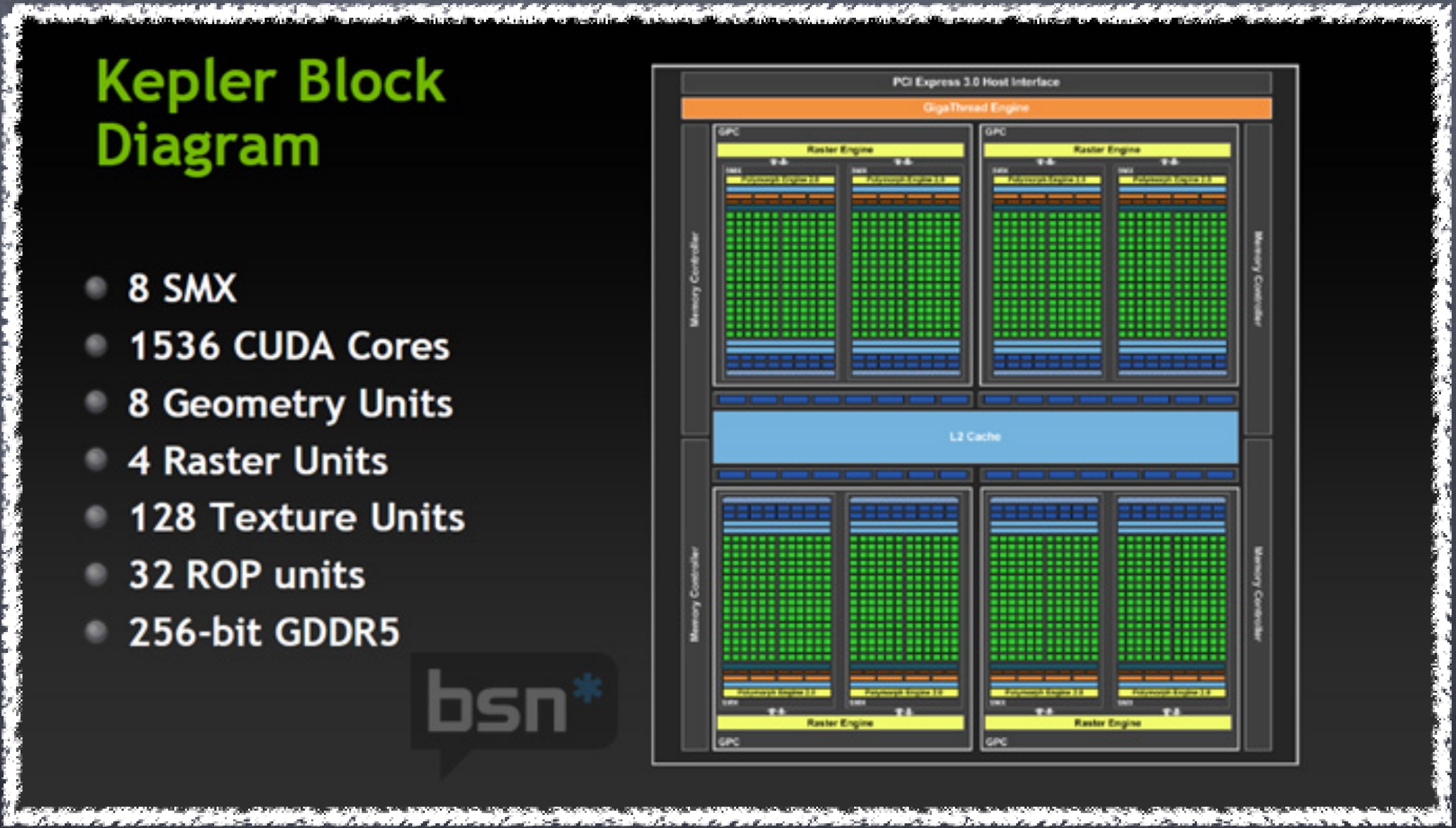


Future plans

Systems	2009	2011	2015	2018
System Peak Flops/s	2 Peta	20 Peta	100-200 Peta	1 Exa
System Memory	0.3 PB	1 PB	5 PB	10 PB
Node Performance	125 GF	200 GF	400 GF	1-10 TF
Node Memory BW	25 GB/s	40 GB/s	100 GB/s	200-400 GB/s
Node Concurrency	12	32	O(100)	O(1000)
Interconnect BW	1.5 GB/s	10 GB/s	25 GB/s	50 GB/s
System Size (Nodes)	18,700	100,000	500,000	O(Million)
Total Concurrency	225,000	3 Million	50 Million	O(Billion)
Storage	15 PB	30 PB	150 PB	300 PB
I/O	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	Days	Days	Days	O(1Day)
Power	6 MW	~10 MW	~10 MW	~20 MW

The present (GPUs)

- Not running Linux
- Different programming paradigm
- Intel is catching up here!
- Fundamental challenge to OS design.



The ugly future

- Exascale supercomputers are planned to have millions to billions of cores.
- Linux must support massive parallelism.
- Fine grained locking is impossible.
- Parallel processing requires for performance in the future.

Conclusion

- Any ideas how to address these issues?

Performance today

- Cache footprint
- Bouncing cachelines
- Atomic operations
- Best performance is a small function that touches a limited amount of memory.