EuroEXA Project

Catania 21 Nov 2017

…why are we here

On the road towards exascale

What is exascale?

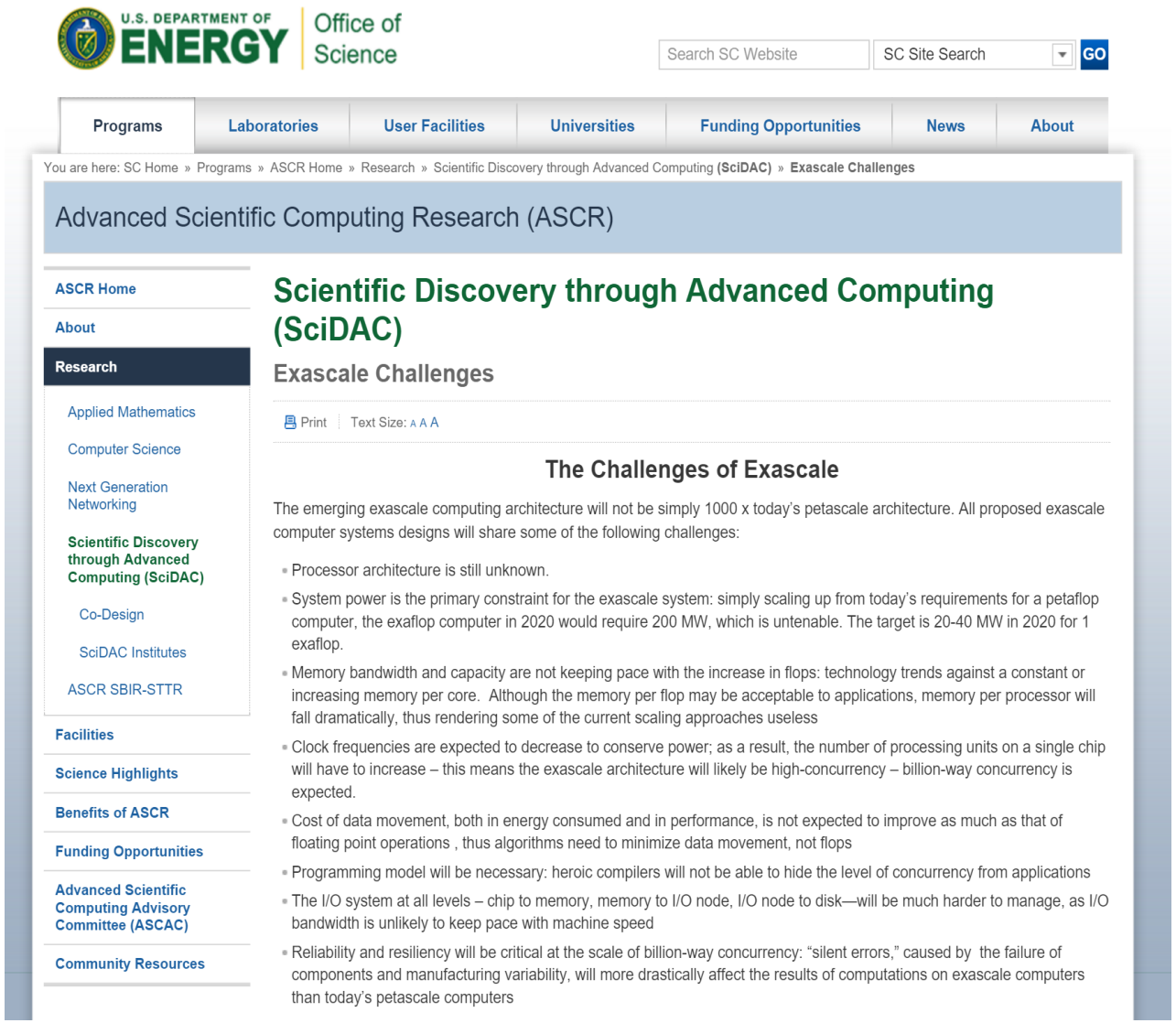
10^{18} floating-point ops / sec

10 to 100 times faster than today's fastest machines
(Human Brain estimated to α ~10^{15}ops/sec)

more than just a peak rate of sustained arithmetic ops
→ 1000-fold better "capability" than that at peta-scale

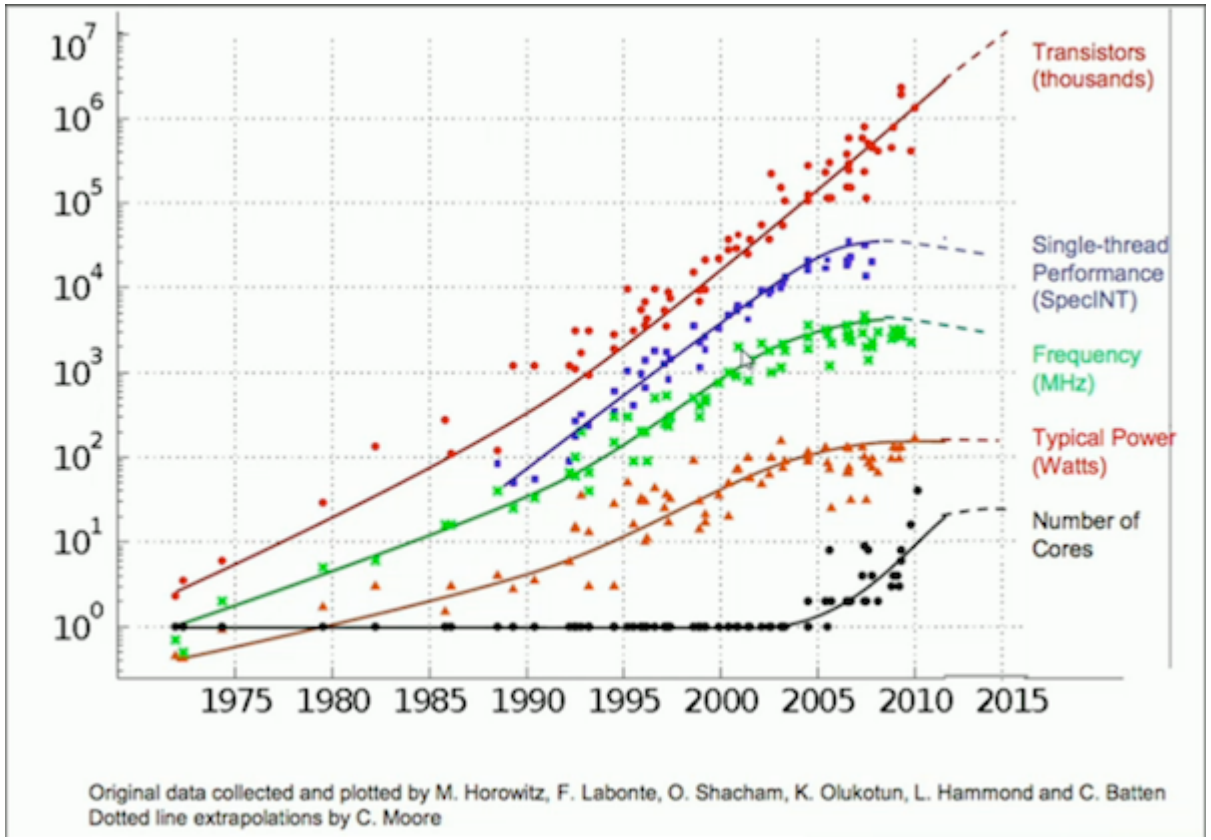
Such a system cannot be produced today.
Its realization is challenging and requires significant advances in a variety of technologies. It is uncertain whether exascale is achievable without disruptive changes in the way we build and use computers and write scientific applications.

Targeting ExaScale: Technological Challenge



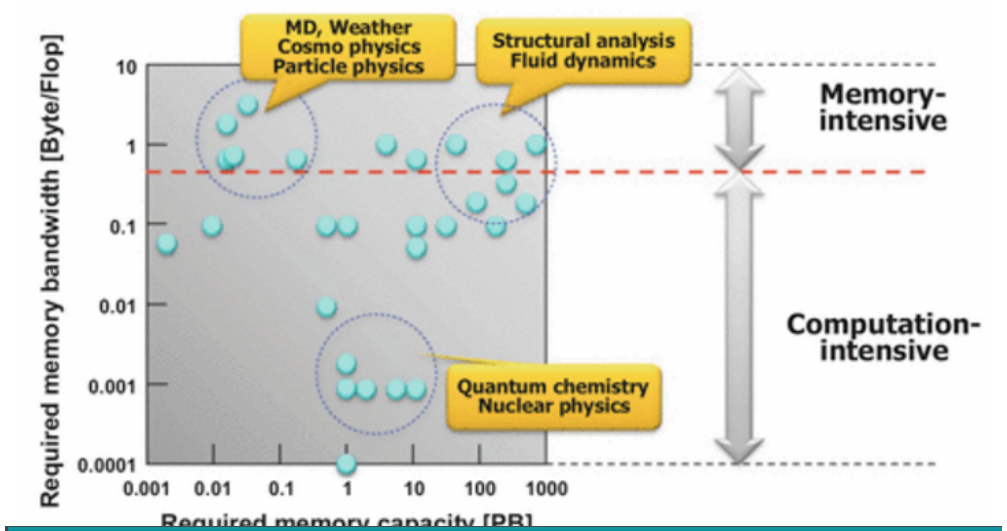
- The Challenge Summary
  - Deliver lots of FLOPS
  - In very little power
  - By 2020
- …the unspoken challenge
  - Is it even feasible using existing computing paradigms ?
  - Other than a couple of governments, who can afford to build one ?
  - How will software use it ?
  - ..Is "Flops" the way to measure it ?

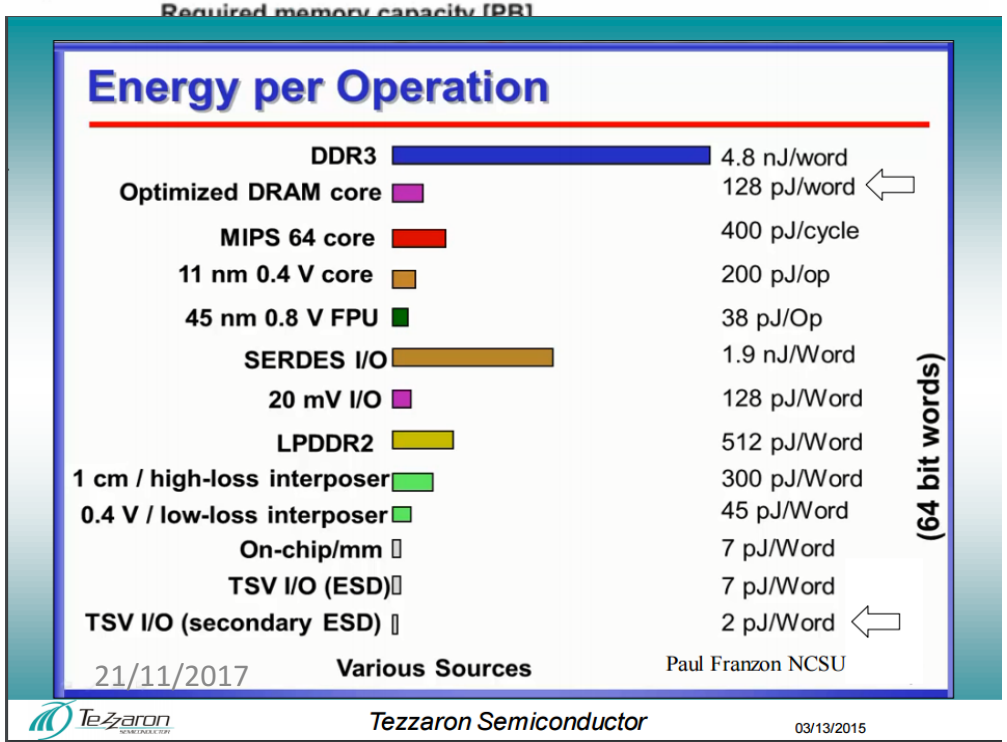
Is many-core the solution ?



- Since 2005, CPU “complexity” reached a plateau
  - No more MHz
  - No more issue width
  - No more power available
  - No more space to add “pins”
- But still get more transistors
  - Current efforts to increase number processor
  - …but

How bad is the memory bottleneck ?





- If designs needs to assume around 1 per FLOPS per byte accessed
  - 500GFLOP processor needs to keep it fed with 500GB/s of main random access memory
- Today's best DDR is ~100pJ/word
  - So 50pJ/byte, or 50M Watts at 1 flop/byte
  - So, exascale target – BUSTED!
- A few GB of capacity can be placed on chip, to bring this to 5M Watt – excluding any static energy of the memory,
- Will SCM (eg 3DXPT) solve this?

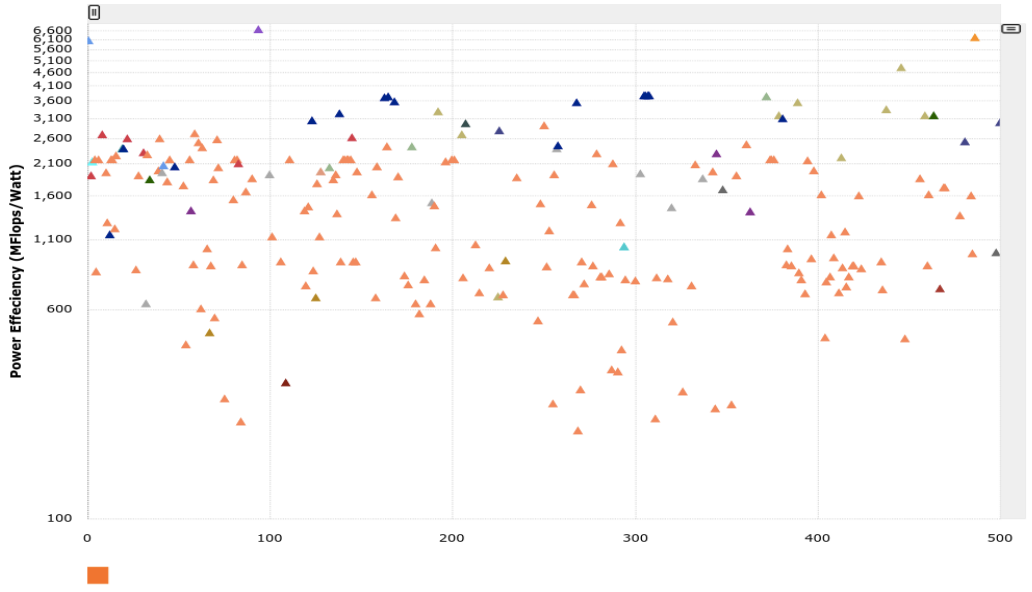
Everting is converted into Energy

Energy Efficiency !!!!!

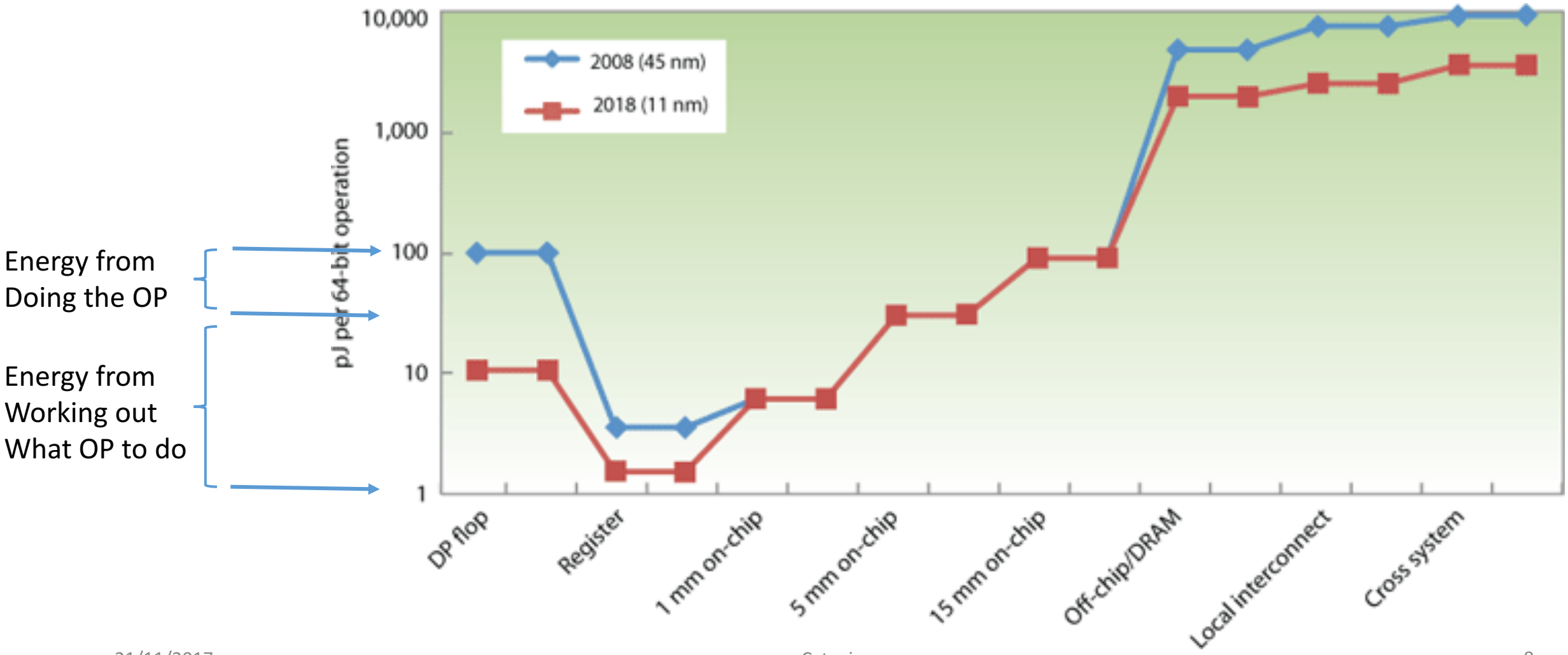
*The goal (constraint) is to sustain an exascale machine with less then **50MW** of power.*

The Chinese Sunway is consuming ~**18MW** (RISC processors).

However, even re-scaling it to Exaflops it would reach **~1.8GW** which is clearly prohibitive.



Energy of data movement operations



Ways to increase processing efficiency

Increase the number of arithmetic operations over the amount of control needed

- Incrementally increase control cost to operate on multiple data items
  - Eg. SIMD or vector machines
- Find a more complex compiler to execute multiple operations in a single instruction
  - Eg VLIW, DSP
- Increase number of control units by reduce their complexity, and operate on multiple data items
  - Eg. GPGPU
- “remove” control, and create a fixed sequence of operations
  - Hardware accelerators
- Consider reconfigurable hardware which enables programmability to execute multiple operations in a single cycle over multiple data items
  - Eg FPGA

Ideally without needing to store intermediate values into a memory (hierarchy)

European efforts: the “Exa” Projects

- EuroServer: Green Computing Node for European microservers
  - UNIMEM address space model among ARM compute nodes
  - Storage and I/O shared among multiple compute nodes
- ExaNoDe: European Exascale processor-memory Node Design
  - ARM-based Chiplets on silicon Interposer
- ECOSCALE: Energy-efficient Heterogeneous Computing at exaSCALE
  - Heterogeneous infrastructure (ARM + FPGAs), programming, runEmes
- ExaNeSt: European Exascale System Interconnect and Storage
  - Interconnect, Storage, Application

…and EuroEXA

- Funded under H2020-EU.1.2.2. FET Proactive (FETHPC-2016-01)
- **€20m** investment over a **42-month** period.
- 16 organizations selected for their technologies and capabilities from across 8 Countries

…and EuroEXA

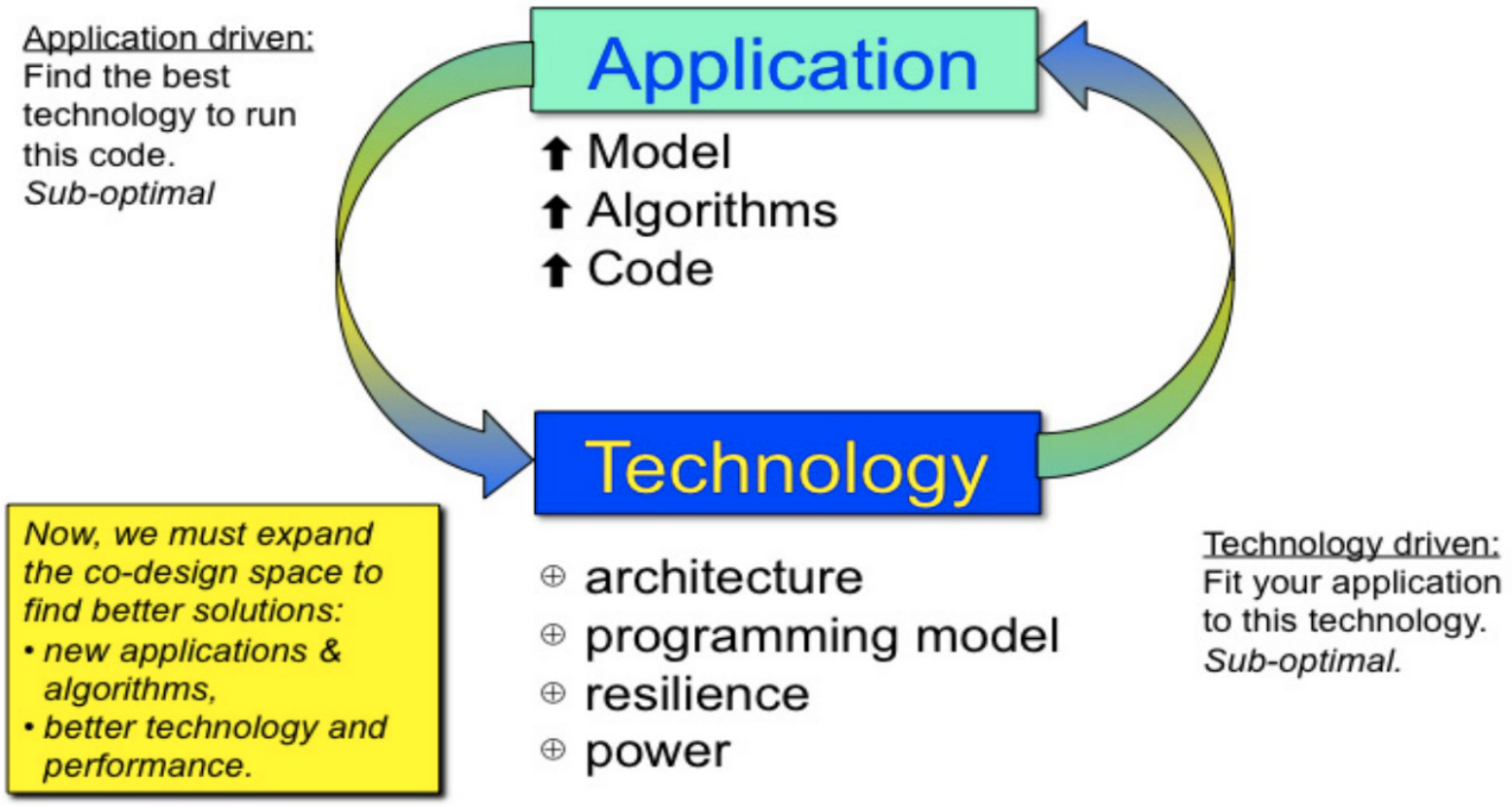
- ARM for control path and communication
  - Exposing *unimem* to help with memory scalability challenges
  - …because we can do that with ARM IP, you can't with a final processor device
- FPGA for acceleration
  - There's lots of DSP in a each FPGA, more than a CPU or GPU
  - There more bandwidth in and out of a FPGA
  - Challenge before us is being about to use lots of them together
- …and today's FPGA have too little ARM vs the amount of FPGA
  - So we'll build a new device that hopefully gives a better balance

Technology approach

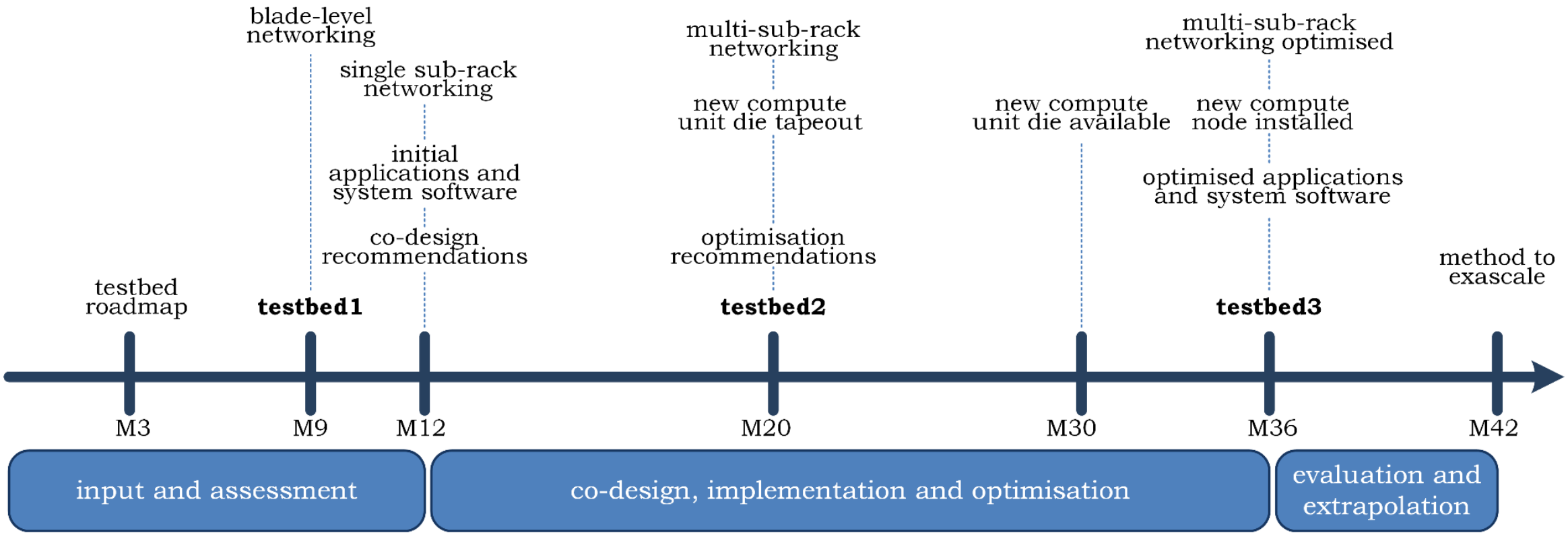
- Use of NVM on RAM sockets (keep storage close to computing)
- Use of Tier approach on parallel filesystem that involves NVM and standard HW
- Accelerators that share RAM with CPUs (FPGA accelerators system software)
- Advanced system software and libraries that hide the complexity of the system

Hardware Software co-design

Applications define the requirements for the system (network, IO, QoS, interconnect, resilience, and more)



Timescale


blade-level networking
single sub-rack networking
initial applications and system software
co-design recommendations
multi-sub-rack networking
new compute unit die tapeout
optimisation recommendations
new compute unit die available
multi-sub-rack networking optimised
new compute node installed
optimised applications and system software
method to exascale
testbed roadmap
testbed1
testbed2
testbed3
M3
M9
M12
M20
M30
M36
M42
input and assessment
co-design, implementation and optimisation
evaluation and extrapolation

EuroEXA: the prototype

- ***Base testbed system configuration*:** 1x Rack, Power and associated cooling comprising 12x Mezzanine boards with local FPGA switch and blade enclosures, initially no switching just pass through to spine. 2x SN2700 switches – half a network group 90x 100Gb cables, providing 6 uplinks from 12 blades, plus 6x3 topology between blades, **72x compute nodes (4x FPGA, 64GB DDR4, 2TB SSD → 1152 Cores, 4TB RAM) .**
- ***At Scale testbed configuration*:** Upgrade of cabinet for full density deployment, 2x SN2700 switches, to support the full 6 sub racks with their 6x fan out into the fat tree spine; 60x additional mezzanine boards with local FPGA switch and blade enclosure (with FPGA providing sub rack level switching); 56x 100Gb cables e.g. 6 uplinks**; 624x Nodes (4x FPGA, 64 GB DDR4, 2 TB SSD) to load all blades to 8x (50%) (2x72 + 8x60 → 10000 Cores, 40TB RAM.**
- A **high performance low latency switching architecture** for the computing nodes of a single Blade (BLES, Blade Level EuroEXA Switch) and a Sub-rack level switching architecture interconnecting all the sub-rack blades with a topology characterised by "geographical addressing"

EuroEXA Boards

- Design and implement ARM subsystem (GPP compute unit)
  - Based on the ARM Buzz subsystem (see next slide)
  - Native Unimem support
  - Dual From/To remote interfaces
    - One to test customer D2D interface with another GPP unit
    - Other to connect to FPGA
- DDR4 memory interface
  - Support ZPT technology
  - Uses Synopsys Ctrl and PHY

System software

"System software must play a more active role in making decisions about how resources are allocated and managed, and the strategies for managing these resources can be very different for different applications."

- System software is dealing with billion-fold concurrency and associated locality.
- System software will have more responsibilities to deliver performance for applications.
- Lightweight specialized Operating Systems to "simplify" access to memory, network and processors.
- New programming models, **runtime environment** and computational libraries

WP3: System Software and Programming Environment

System software and programming environment for the EuroExa node:

- OS and firmware
- **Numerical libraries**
- **Programming models, runtime systems and tools**
- Efficient use of resources
- **Hyperconverged storage**

WP3 structure

Work package number	WP3			Start month and duration			M1 - M36	
Work package title	System software and programming environment							
Participant number	1	2	3	**4**	5	6	7	8
Participant short name	ICCS	UNIMAN	BSC	**FORTH**	STFC	IMEC	ZPT	ICE
PMs per participant	22	62	94	**88**	18	0	4	0
Participant number	9	10	11	12	13	14	15	16
Participant short name	ALLIN	SYN	MAX	NEUR	INFN	INAF	ECMWF	FRAUN
PMs per participant	14	28	94	11	24	13	0	37

Hyperconverged storage software for extreme data applications

- Participants: FRAUN, INAF
  - FRAUN: Data locality support in BeeGFS, support for placement rules
  - INAF: Requirements extraction from big data applications, testing, evaluation

- INAF is the only big data “provider” of EuroEXA!!!!

- MIRIAD or other SKA precursors…

Programming model and runtime support

- INAF will work on checking and testing functionalities of major numerical libraries used by its codes, and collects requirements from WP2 in terms of software libraries.

- INAF will work on preliminary profiling and assessment of functionality and performance of the EuroEXA programming model runtimes.

- Testbed integration, firmware and bring up

Exascale Applications

- New algorithms must take into account communication/synchronization - avoiding algorithms that increase the computation/communication ratio (Flops per communicated Bytes
- algorithms that implement a law B/F ration (<0.1)
- algorithms that support simultaneous computation/communication,
- algorithms that vectorize well and have a large volume of functional parallelism.
- algorithms that adaptively respond do load imbalance of billion-threads scale (e.g. dynamic scheduling by DAG) without compromising with spatial locality

WP2: Applications

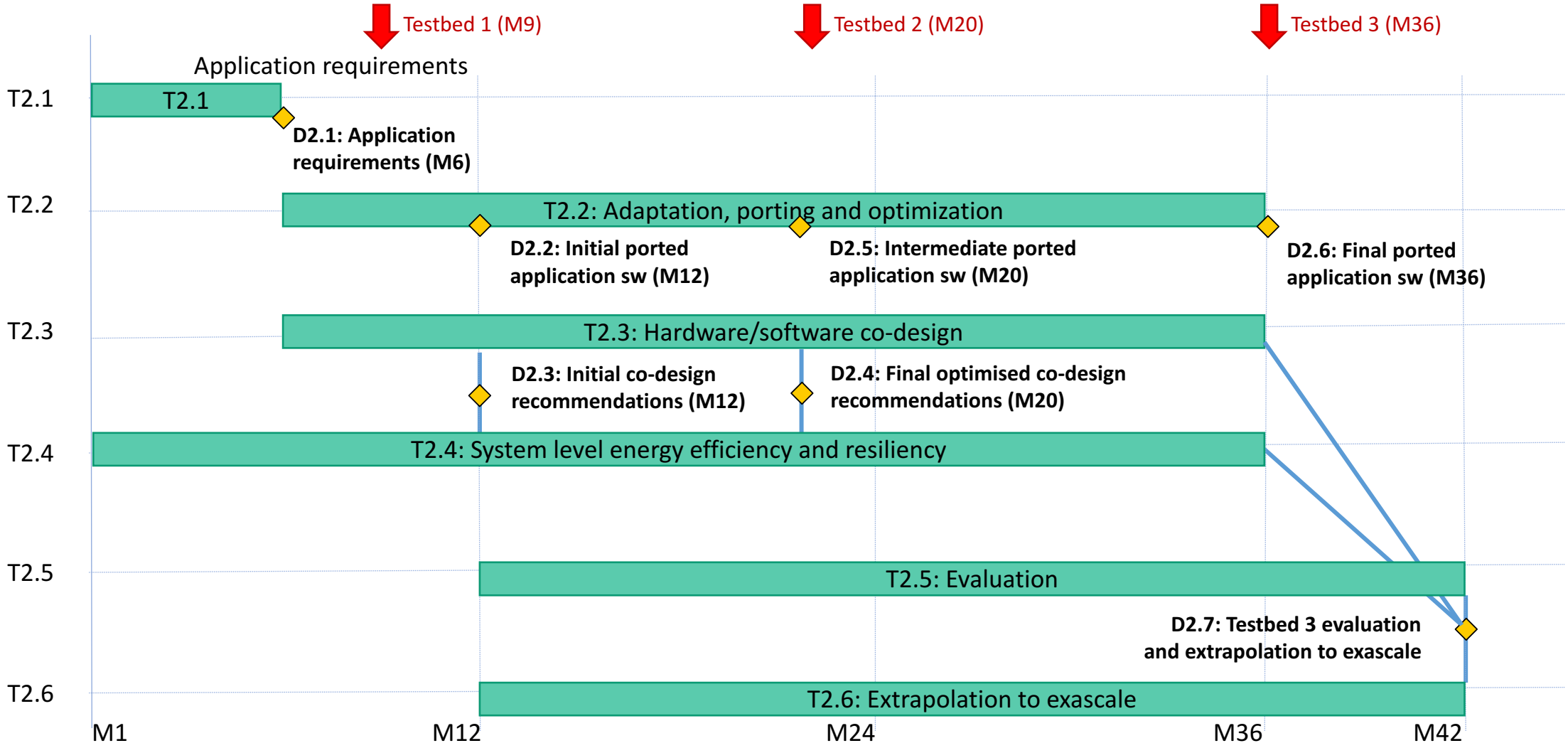
- To quantify and analyse the **hardware requirements of the HPC applications** that will be used to co-design the EuroEXA architecture through adapting, porting and running codes on Testbeds 1 and 2.
- To recommend the **co-design**ed balanced architecture for the EuroEXA Testbed 3.
- To **adapt, port and optimise** the HPC applications to take full advantage of the EuroEXA Testbed 3.
- To understand and quantify the HPC **resiliency** [and energy] problem from the whole system perspective.
- To **evaluate** the performance, scalability, energy efficiency and resiliency of the three Testbeds.
- To **extrapolate** from the evaluation of Testbed 3 to the performance of a future exascale system.

WP2 organization

- All partners involved
- 541 PMs (the biggest WP)
- 7 deliverables

Work package number	WP2			**Start month and duration**			M1 – M42	
Work package title	Applications, co-design, porting and evaluation							
Participant number	1	2	**3**	4	5	6	7	8
Participant short name	ICCS	UNIMAN	**BSC**	FORTH	STFC	IMEC	ZPT	ICE
PMs per participant	68	24	**92**	29	36	36	3	4
Participant number	9	10	11	12	13	14	15	16
Participant short name	ALLIN	SYN	MAX	NEUR	INFN	INAF	ECMWF	FRAUN
PMs per participant	12	35	6	40	38	48	39	31

WP2 tasks and deliverables



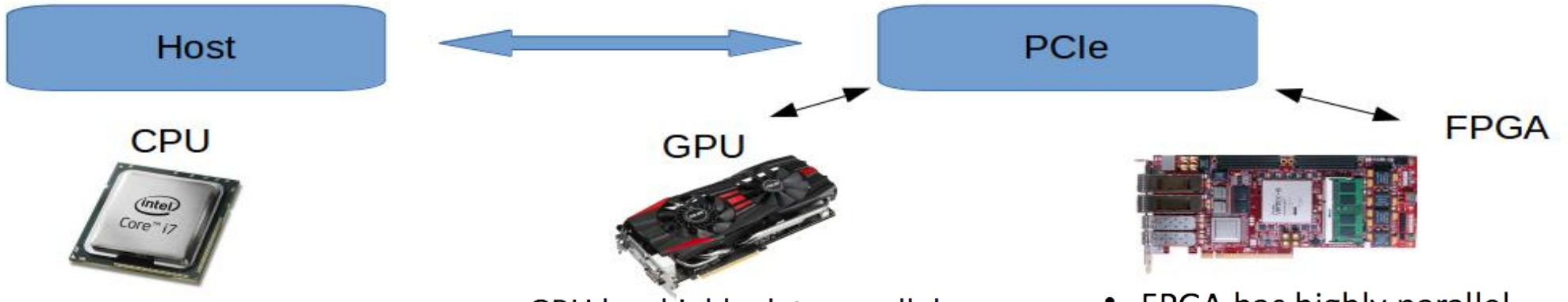
WP2 main tasks

- Define statistics of interest
  - e.g. memory footprints, memory/network latencies and bandwidths
- Profile applications on partners' HPC machines
  - Also on Testbed 1 (due M9 after end of T2.1)
  - These measurements need to feed into T2.3 on co-design
- Identify use cases for UNIMEM and/or FPGA
- Port and optimize applications on Testbeds
- Recommend architecture for EuroEXA Testbed 3
  - Mixture of general purpose and FPGA
  - Memory capacities and bandwidths
  - **I/O interfaces and bandwidths**
- **Extrapolate** real-application performance to exascale
  - Key requirement in the call text

Astrophysical codes

- HiGPU
- PINOCCHIO
- N-Body Simulations
  - GADGET (http://wwwmpa.mpa-garching.mpg.de/gadget/)
- GAIA:
  - AVU-GSR is a matrix solver for extremely large sparse matrices
- SKA precursors big data applications

Power efficient acceleration needed



- CPU is latency-optimized (each thread runs as fast as possible, but only few threads)
- CPU has few cores (≤ 16)
- CPU excels at irregular control-intensive work (lots of hardware of control, fews ALUs)
- Programming languages: C/C++, Fortran, Python, IDL, ...
- Parallel libraries/directives: MPI, OpenMP, ...

- GPU has highly data-parallel fixed architecture (SIMD)
- GPU is throughput-optimized (thousands of threads)
- GPU excels at regular math-intensive work (lots of ALUs for math, little hardware control)
- Very high memory bandwidth (drawback for power consumption)
- Parallel programming: OpenACC (directives), CUDA, OpenCL (low level programming required for high performance)
- GPU has short life cycle

- FPGA has highly parallel customizable architecture (both data or task parallel computation)
- On FPGA functionality can be re-programmed by downloading a configuration into the device
- FPGA has optimal power efficiency (tipically 1/5th that of GPU)
- Parallel programming: OpenCL (low level programming required for high performance)
- FPGA has long (> 15 years) life cycle

The astrophysical codes in ExaNeSt: *HiGPUs*

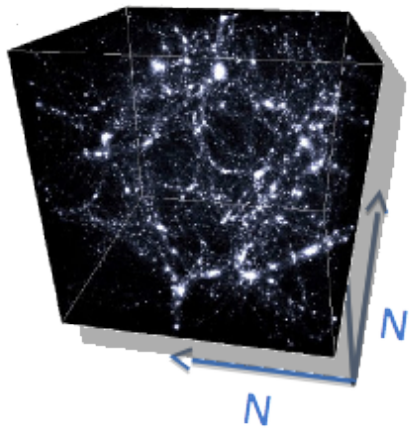
```
  [...]
  /* Loop over blocks */
  for (unsigned int i=0 ; i<particles ; i+=blockDim)
    {
//#pragma HLS UNROLL factor=4
      int address = i + threadIdx + costante2 + istart;

      shPos[threadIdx] = globalX[address];
      shVel[threadIdx] = globalV[address];
      shAcc[threadIdx] = globalA[address];

      barrier(CLK_LOCAL_MEM_FENCE);

      /* Loop over particles within the block */
      for (unsigned int j=0 ; j<blockDim ; j++)
        {
#pragma HLS PIPELINE
          float4 dr = convert_float4(shPos[j] - myPosition); dr.w = 0.0f;
          float distance = dot(dr, dr) + EPS*EPS;
          float sqrdist = convert_float(shPos[j].w) * native_rsqrt(pown(distance,3));
          float rdistance = native_recip(distance);

          float4 dv = shVel[j] - myVelocity; dv.w = 0.0f;
          float4 da = shAcc[j] - myAccelera;

          float alpha = dot(dv, dr) * rdistance;
          float beta = -3.0f * sqrdist * ((dot(dv, dv) + dot(dr, da)) * rdistance + pown(alpha,2));

          float4 au = (sqrdist * dv) + (-3.0f * alpha * sqrdist * dr);

          acc += convert_double4(sqrdist * dr);
          jrk += convert_double4(au);
          snp += convert_double4((sqrdist * da) + (beta * dr) + (-6.0f * alpha * au));
        } /* End of loop within the block */
      barrier(CLK_LOCAL_MEM_FENCE);
    } /* End of loop over particles */
  [...]
```

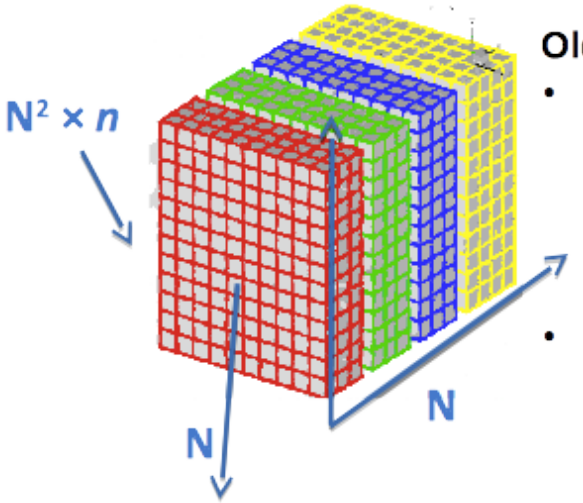
- ***Kernel extracted from HiGPUs***
- ***Kernel re-designed for FPGA:***
  - ***vectorization*** *(each work-item does 4x much work)*
  - ***geometric functions added***
  - ***usage of the local memory of the FPGA***
  - ***emulated double-precision arithmetic adopted*** *(currently DP arithmetic is resource-eager and performance-poor in FPGAs)*
- ***Working with eXactLAB® and ECOSCALE to download a configuration into the FPGA*** *(difficult task)*

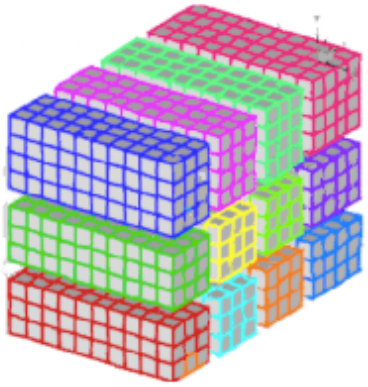
Lesson learned:

- on GPU you seek the best mapping of the algorithm onto a fixed architecture;
- on FPGA you seek the best architecture for an algorithm while designing the kernel (hardware features matter) and to balance the throughput and resource usage (high performance per watt solutions)

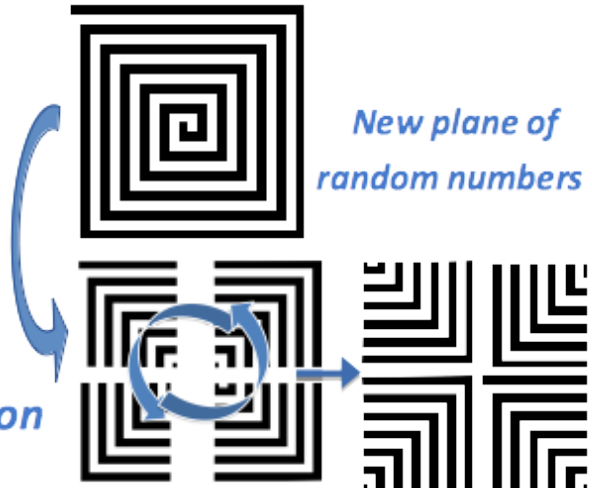
PINOCCHIO and GenIC







2D-3D FFT decomposition



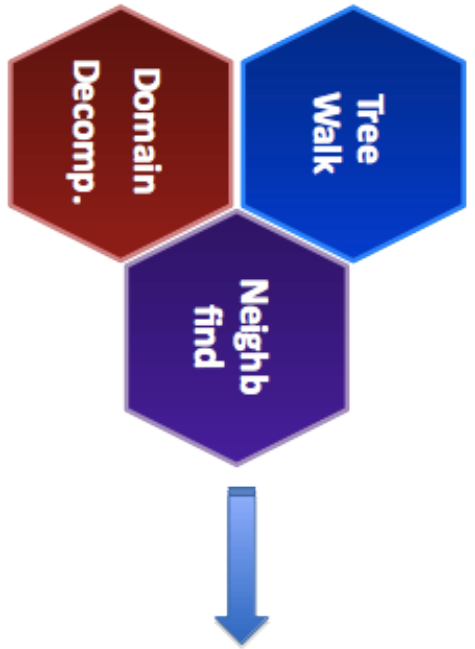
Old version of PINOCCHIO:

- use of FFT(W) with 1D spatial decomposition;
  - *N* calculating task at most;
  - memory limitation when *N* becomes really large (mass and spatial resolution are related with the grid number *N*);
- initial power spectrum has physical properties and symmetries and it's built from a plane of random numbers that is entirely replicated among MPI tasks, posing severe memory limitations when *N* is large (several 10^4 or more).

Re-engineered version of PINOCCHIO:

- use of FFT(W) with 2D-3D spatial decomposition;
- re-designed algorithm to generate power-spectrum;
  - it has the same properties and symmetries;
  - each MPI task has only its portion of the pseudo-random field;
- [ongoing] detailed analysis of memory pattern and access

GADGET: complete re-design



Extremely complex code (about 200k lines):

- many physical processes and diverse algorithms;
- rigid procedural design (MPI tasks handle global operation blocks).

Jumping to exa-scale codes requires:

- take into account the **NUMA hierarchy**;
- re-design algorithms in a **task-based**, **data-driven** perspective.

[ongoing] working on kernels for:

- tree walk;
- neighbors finding;
- domain decomposition.

Resilience

How many hardware failures each day? Try to guess....

- Impact on interconnect design (routing and scalability)
- Impact on Filesystem (availability and scalability)
- Impact on System Software (availability and scalability)
- Impact HW design (packaging and topology)
- Virtualization for/in HPC

- Impact on Applications: is check-pointing mechanism (terascale era approach) a valuable methodology?
  - Not all A&A applications are using CP
  - CP relies on FS (do you remember the data movement problem?) and its availability
  - System Software must be aware of CP and restart (queue systems)

Conclusions

- We are in the way towards Exascale
- This is not only an HPC business but also HPDA
- Developing expertise in new HPC and HPDA software and infrastructures.
- Developing expertise on new devices (FPGAs as computing resource)
- Great effort in redesign codes and algorithms…the prima we do the better it is.

- We are in an excellent position shuld keep this avatange for the future:
  - We have an open position for SW developer
  - We need to invest more and improve internal coordination