ETP 4 HPC International Collaboration

European Commission

Exploiting the Potential of European HPC Stakeholders in Extreme-Scale Demonstrators

Top-Down System Design Approach

Hans-Christian Hoppe, Intel Deutschland GmbH

e×dci European Extreme Data & Computing Initiative

European HPC Summit Week 2017, Barcelona

Motivation & Introduction

Computer system design requires a fair amount of top-down thinking

- Performance characteristics and tradeoffs
- System architecture

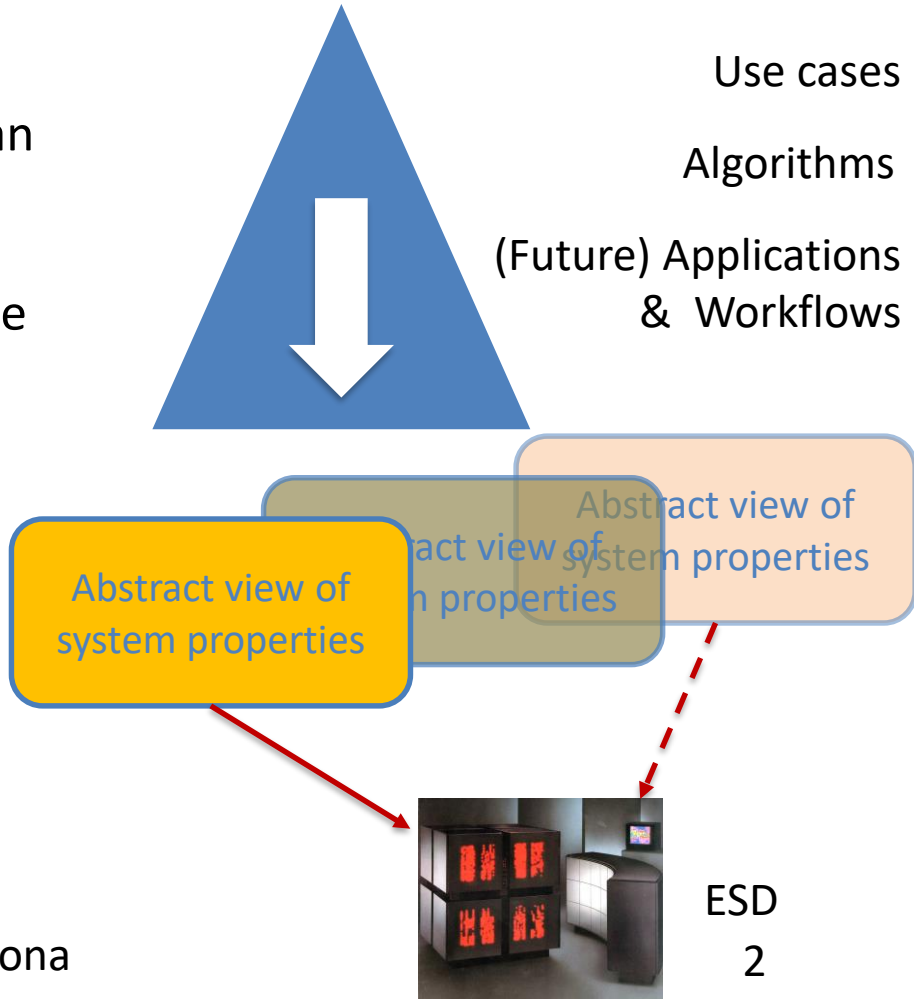
EsDs must match the need of relevant, scientific or industrial European use cases

It's only logical to drive the top-down design effort from the actual use cases

- Collect relevant use cases & algorithms
- Analyze *future* requirements and characteristics
- Create an abstract view of application & required system properties
- Cluster these into a suitable number of design points (= EsDs)

At the end, a careful judgement call is needed to define the scope that an EsD should support

- One use case vs. many use cases vs. all use cases



Data Collection - Three Steps

High-level application properties

Use cases → algorithms→ applications

- Data structures
- Algorithms (compute & data)
- Scaling and parallelism
- Usage model

Quantifiable system properties

- Compute performance (FP, integer, …)
- Memory size and performance
- Interconnect performance
- I/O volumes
- Energy efficiency

Non-quantifiable system properties

- Communication & I/O structures,
- Memory access patterns
- Communication patterns
- I/O patterns
- Heterogeneity
- Programming & execution models

General Remarks

It is imperative to look at use cases and applications relevant at deployment of the EsD

- We need to address tomorrow's problems, not today's

The selection of use cases to be supported is critical

- It will shape the architecture
- Danger of being too narrow vs. too broad
- Who decides on this? (end-users, OEMs, HPC centers, Brussels, …)

System architecture & design is always a tradeoff

- Need to have specific & complete properties / requirements

Accelerators & Heterogeneity

The need for “accelerators” is a result of application characteristics / requirements

- Like a combination of highest memory BW, small memory footprint and and very wide SIMD-style computation

The need for heterogeneity can arise at three levels

- Different parts of your application have (quite) different characteristics
- Different applications that make up one use case have different characteristics
- One system has to support multiple use cases of different characteristics

The “best” system architecture will likely be different for each of these three cases …

Energy & Energy Efficiency

The combination of use case(s) and operator will define the exact power requirements

- Power used depends on the application(s), and different centers can accommodate different power ratings

General power efficiency requirements do not make too much sense (like Flop/s/W)

- It depends on the application and it's use of the system
- Need more detailed power metrics, or stick with specific applications

The full energy use of a system has to be factored in here

- CPU + memory + interconnect + storage + PDUs + cooling

Wrapping it All Together …

We should have a good initial picture of the scientific use cases / applications

- Through the CoEs and the FETHPC application activities

We do have certain insight into important commercial/industrial applications

We do have the operators of future large HPC systems in the discussion

- HPC computer centers know their workload mix and operational requirements

We can progress with top-down design in one of two ways

- Global: create a shared repository of system properties across the European use cases, let EsDs chose which ones to design for
- Local: each EsD selects the use cases and does all of the analysis/data collection

ETP 4 HPC International Collaboration

European Commission

Backup

Use Case/Application Information

Data structures: n-D structured or unstructured meshes, trees, graphs, particles, adaptive mesh refinement

Compute algorithms used: dense or sparse LA, particles & fields, n-body methods, spectral methods, multiscale methods, explicit vs. implicit solvers

Data algorithms used: streaming vs. “in place”, search/query/indexing, ML methods used/planned, ratio of operations vs. transactions

Parallelism: embarrassingly parallel, bulk synchronous, dataflow/workflow, map-reduce (general)

Specific, Quantifiable System Properties

Delivered computational performance (by thread, process or complete system)
- FP instructions/sec (SP, DP), INT instructions/sec

Memory system performance (by thread, process or node)
- Sustained memory bandwidth (GByte/sec)
- Memory size per thread/process (GByte)
- Compute vs. memory (Flop/Byte ratio)
- Memory access and reuse patterns

Interconnect performance
- Point-2-point latency (µsec), bandwidth (GByte/sec)
- Bisectional bandwidth (Gbyte/sec)
- Compute vs. interconnect (Flop/Byte ratio)
- Communication patterns

I/O performance
- Read and write latencies (µsec) and bandwidths (GByte/sec)
- I/O volumes for full application and over time (GByte), bursts (GByte/sec)
- I/O activity due to resiliency (e.g. checkpoints, rates and volumes)

System scale
- Delivered full system performance (EsD and Exascale timeframe)
- Realistic degree of parallelism (# of threads and processes, size of shared-memory islands)

Energy
- Energy to solution (Wh), maximum power use (W), energy per operations (W/Flop, W/Byte, …)

General Non-Quantifiable System Properties

Programming models: OpenMP, task-based dynamic threading, MPI, PGAS, distributed data flow, …

Execution models: batch single-step, workflow multi-step, interactive viz/control, malleable execution

Resiliency:

Communication structures: n-D halo exchanges, any-to-any communication, reductions, scatter-gather

I/O structures:

Memory access patterns (spatial & temporal): contiguous/strided/indexed, data reuse, repeating address streams

Heterogeneity: steps/phases in the application with different requirements (please provide separate category #1 and #2 data for each)