PDC Center for
High Performance Computing

Collaborative Research into Exascale Systemware, Tools and Applications

Erwin Laure

Director PDC-HPC
KTH, Sweden

Based on material from Mark Parsons, Cresta Project Director, EPCC

Outline

- The exascale challenge
- The CRESTA project
- The CRESTA approach to exascale

Current Challenges in Supercomputing

- We are at a complex juncture in the history of supercomputing
- For the past 20 years supercomputing has “hitched a lift” on the microprocessor revolution driven by the PC
- Hardware has been surprisingly stable
- The programming models for these systems were very similar
- But now we have a problem ...

Lindgren at PDC

- 16 cabinet Cray XE6, 36,384 cores, 305 TF TPP
- 2 x12 core AMD Opteron 2.1 GHz CPUs, 32 GB RAM

KTH VETENSKAP OCH KONST

CREST

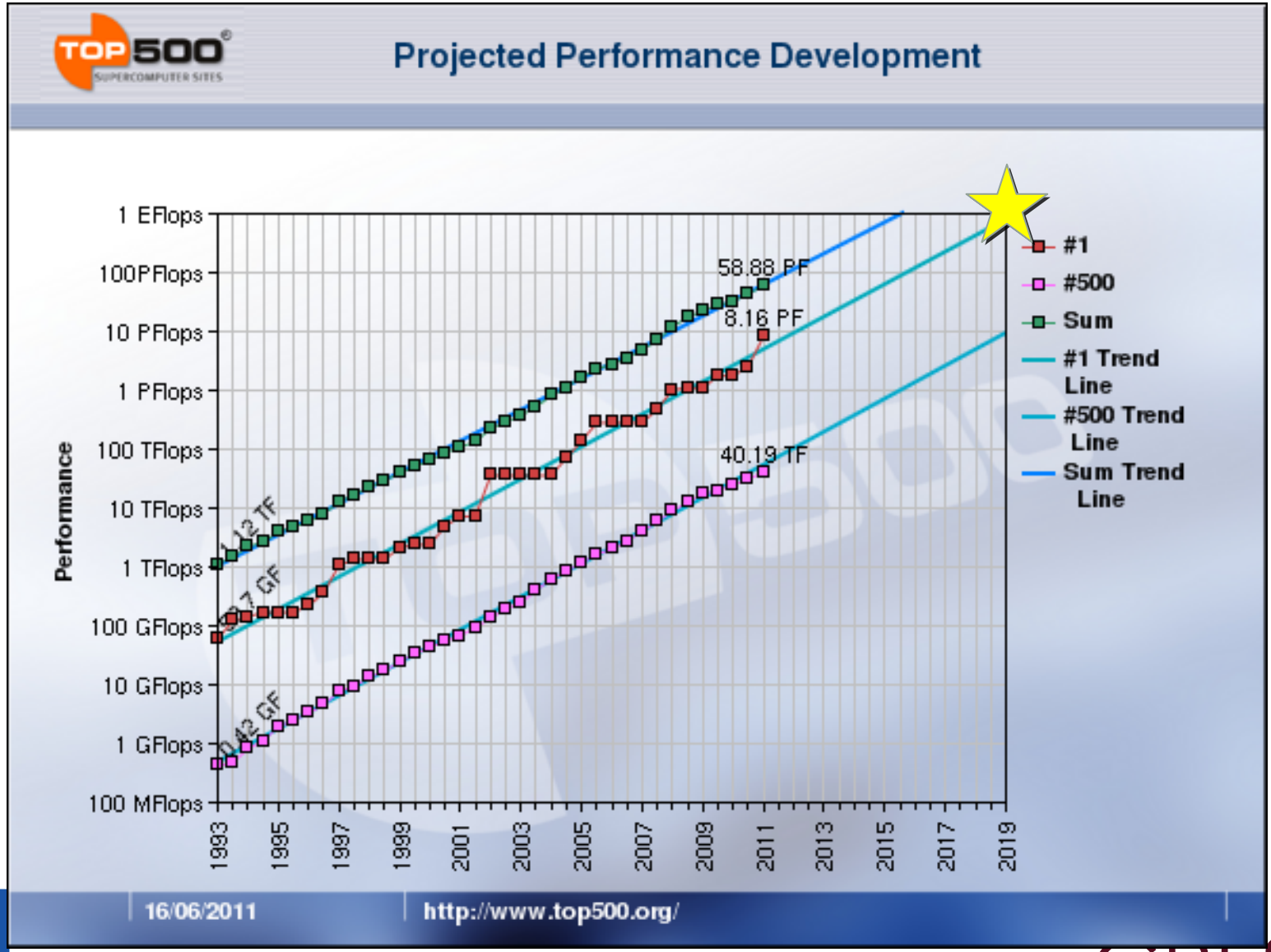
The many-core future

- Hardware is leaving many HPC users and codes behind
  - Clock rate is going down, number of cores is increasing
  - Memory per core is going down
- Majority of codes scale to less than 512 cores
  - These will soon be desk-side systems
- Less than 10 codes in EU today will scale on capability systems with 100,000+ cores
  - Lindgren already has more than 36,000 cores
  - Germany's Jugene system already has 294,912 cores
- Many industrial codes scale very poorly – some codes will soon find a laptop processor a challenge!
- Much hope is pinned on accelerator technology
  - But this has its own set of parallelism and programming challenges
  - Many porting projects to GPGPU have taken *much* longer than expected

Where next for supercomputing?

- Currently we are in the single petaflop age
  - Rank 1 in Top 500 June 2011 has 8 PF
- Systems today are already very difficult to program and run
  - 100s of thousands of cores (K: 540k)
  - Massive power requirements
    - Today we use around 3 MW (K: 10 MW)
  - Many codes simply don't scale beyond a few thousand cores
- And yet we're talking about how to reach exascale by 2018/19 …

Shooting for an exaflop



(With thanks to top500.org)

What are the challenges?

- DARPA conducted a study on exascale hardware in 2007
- Objective: understand the course of mainstream technology and determine the primary challenges to reaching 1 Exaflop by 2015, or soon thereafter
- They concluded the four key challenges were:
  - Power consumption
  - Memory and storage
  - Application scalability
  - Resiliency
- See
  - http://www.darpa.mil/ipto/personnel/docs/ExaScale_Study_Initial.pdf

1: The power problem

- The most power-efficient microprocessors available today deliver ~450 Mflops/W on Linpack (K: 824 Mflops/W)
  - ie ~2.2 MW per petaflop/s … or 2.2GW per exaflop/s
  - Excluding cooling which adds 20-100% to the power draw
- … clearly, we have to do better!
  - DARPA goal: 50 Gflops/W in 8 years
  - 100x improvement
- But even then
  - That still equates to a 20MW computer
  - A number of US labs are currently putting in 30-40MW machine room power supplies

Forsmark: 3.2 GW

Memory and power

- Memory bandwidth has increased ~10x over the past decade
- The energy cost/bit transferred has declined by 2.5x
- … energy cost of driving the memory at full bandwidth has risen 4x
- Memory DIMMs can't provide bandwidth at acceptable energy costs
- And today's applications use more memory than ever before

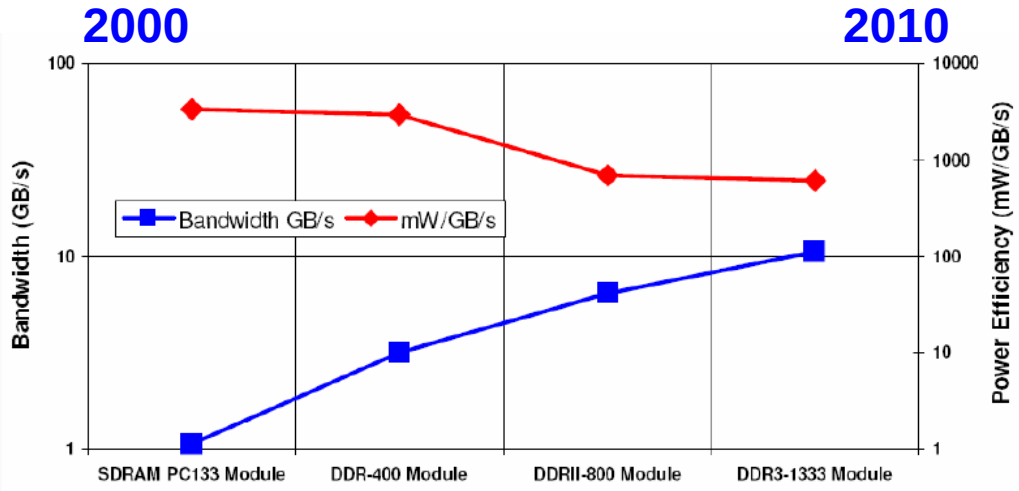
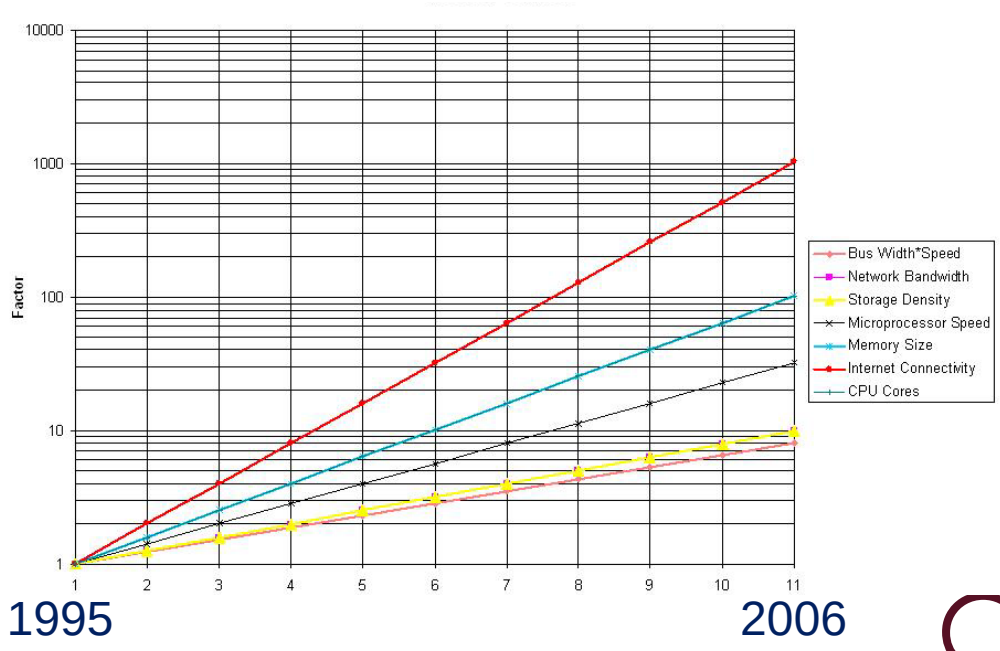


Figure 6.22: Commodity DRAM module power efficiency as a function of bandwidth.

2: Memory performance

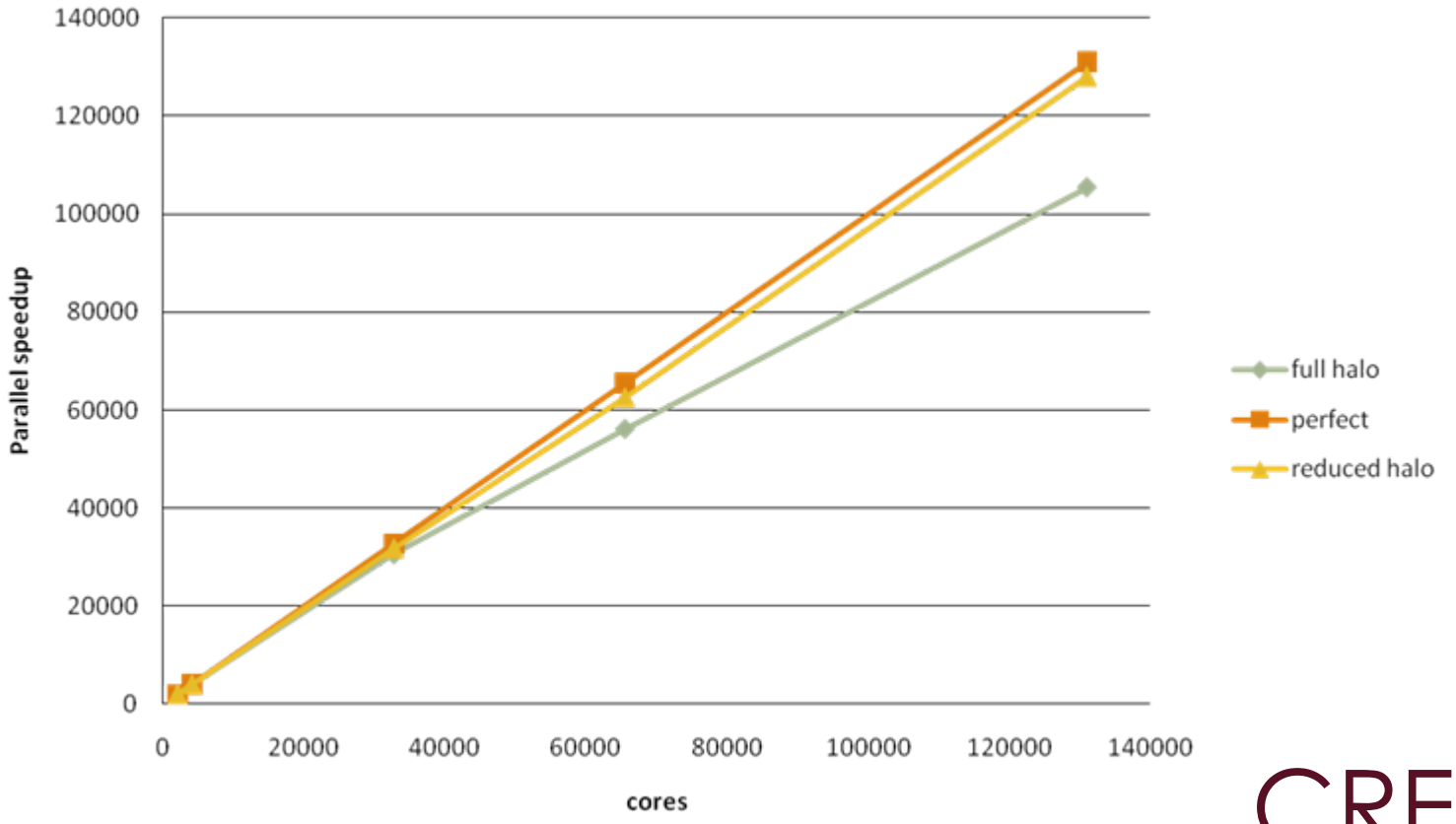
- Over the past 30 years DRAM density has increased ~75x faster than bandwidth
- … memory bandwidth is the limiting factor in future designs
- Novel memory technologies needed :
  - phase-change memory, holographic memory, graphene …



3: Applications scalability

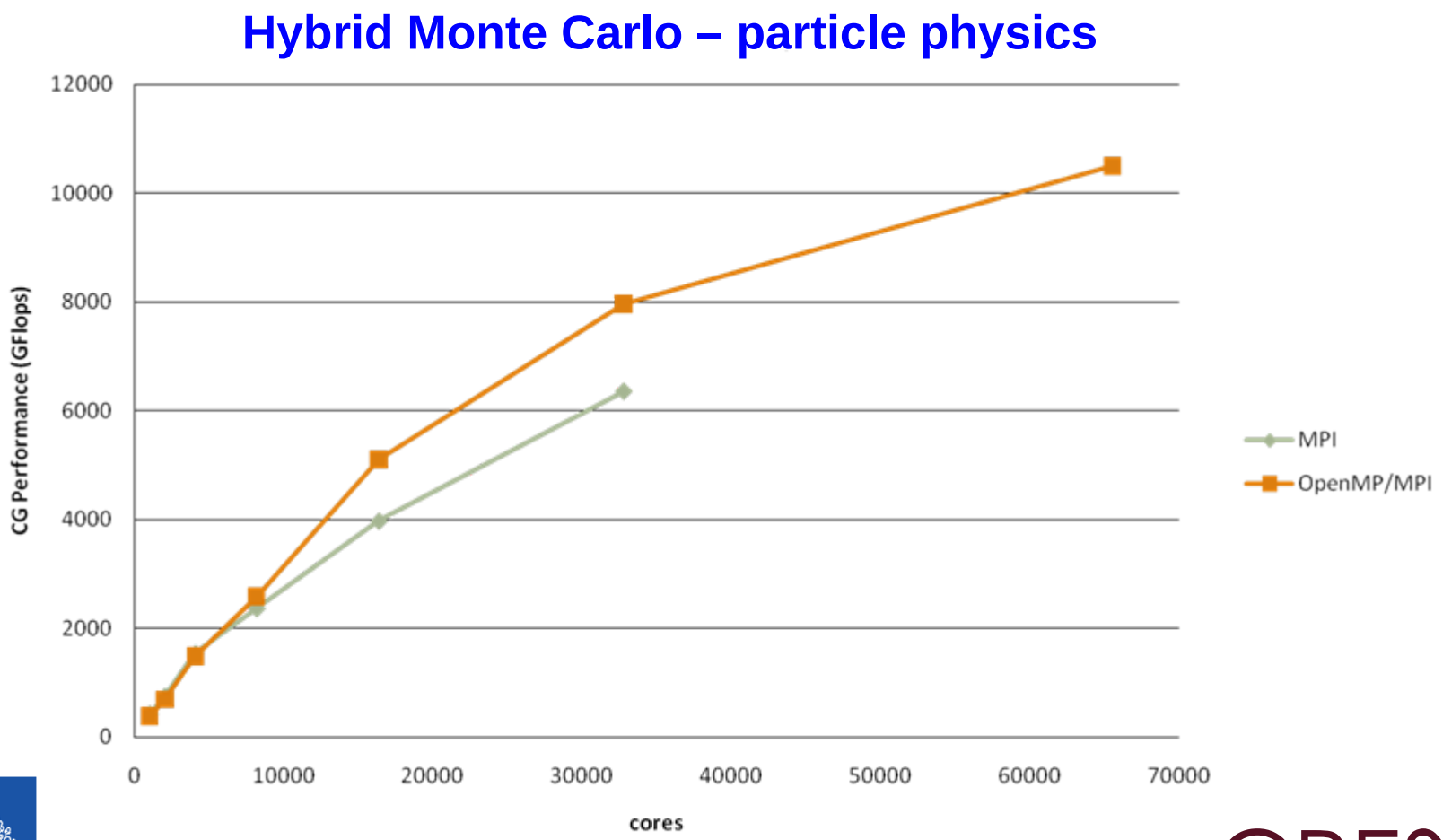
- Those codes with low communications overheads and which can exploit weak scaling do well:

  Lattice Boltzmann – soft condensed matter



3: Applications scalability (cont)

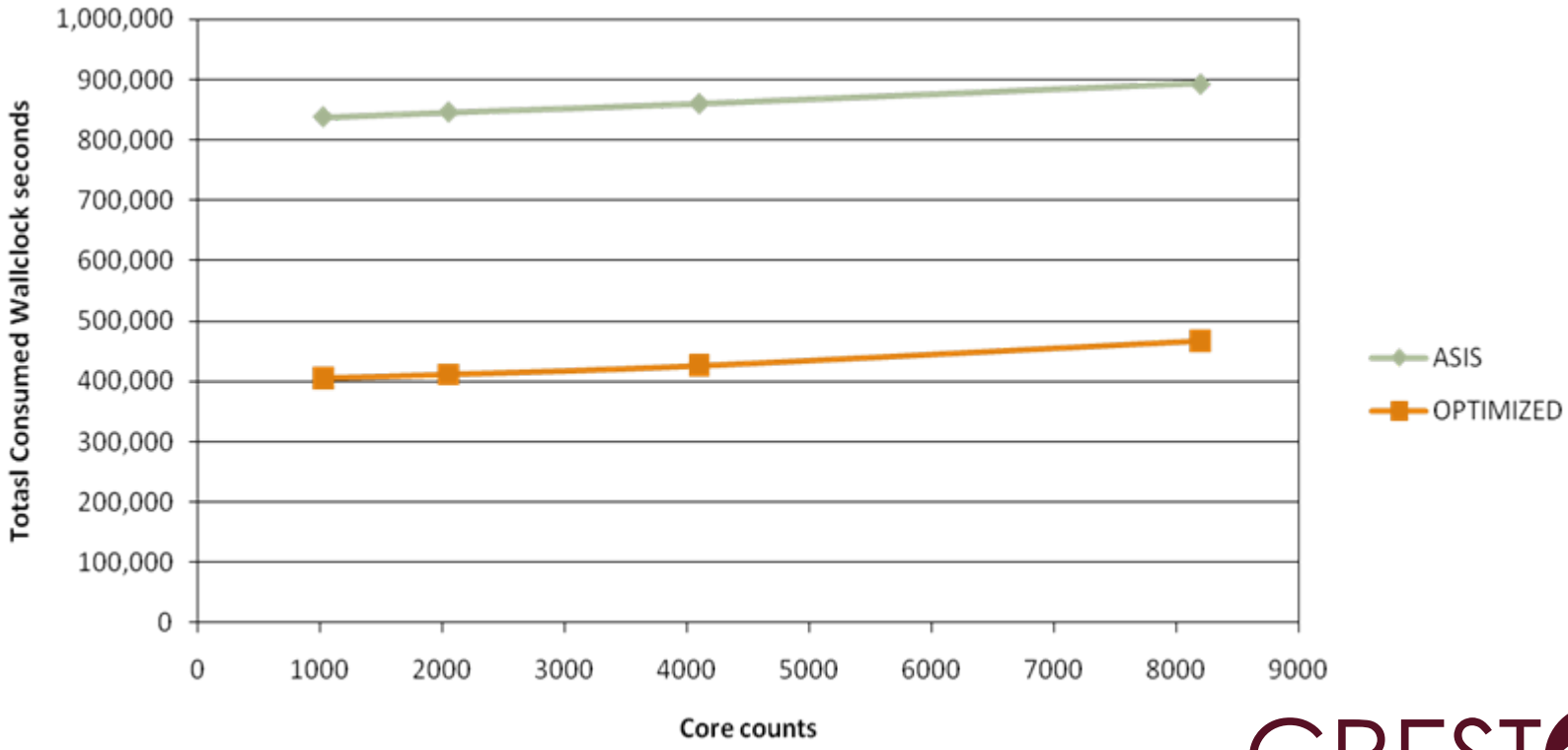
- … some do pretty well



3: Applications scalability (cont)

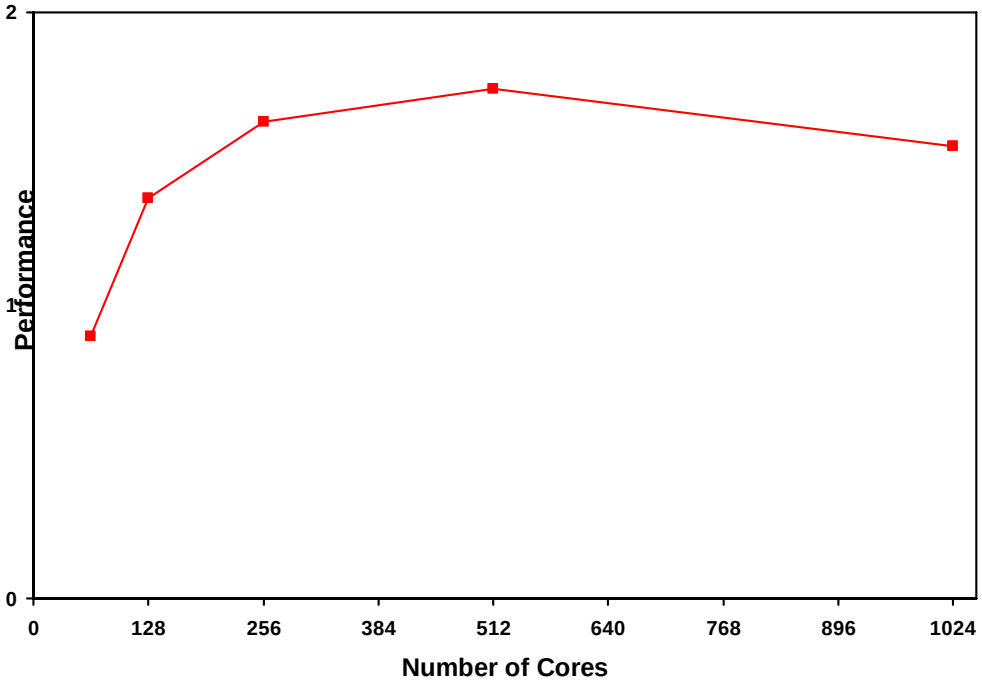
- … but most are disappointing
  - this behaviour is caused by the overheads of global communications
  - Applications scale only when communications are highly infrequent, or local

Lattice Boltzmann – biophysics



3: Applications scalability (cont)

- Users, especially in chemistry and engineering, are locked-in to poorly-scaling third-party codes
- Summary: widespread need for good software engineering and parallel techniques
- These are very bad results …
- And surprisingly common



3. Applications scalability summary

- Strong versus weak scaling
  - Weak scaling (problem size varies with machine concurrency) has been the mainstay of parallelism for 30 years
  - Strong scaling (scaling with a fixed problem size) has been hard to find
- For some applications there is no more weak scaling because the system being studied is already large enough
  - Example: classical molecular dynamics for many chemistry applications only requires 100 - 1000 molecules
- An even larger set is constrained by algorithmic complexity
  - There is simply not enough concurrency in the algorithm
  - Modern hardware – multicore and GPGPUs – are cruelly exposing this
- The numerical core (and probably much more) of many applications will have to be rewritten to achieve exascale performance

4: Resiliency

- An Exaflop machine is likely to have ~1 million processors
- If each processor had a lifetime of 10 years (unlikely)
- … then the machine will have a MTBF of ~5 minutes!
- We therefore have to be able to operate it in a way which is resilient to single-node failures
- Unfortunately, most scientific applications use synchronous algorithms
- … which would halt when something blocks the data flows
- Fault tolerance is not a new problem
  - von Neumann considered this in detail as early computers failed often
- Much work is being done today in this area

International Exascale Software Project

- IESP is a CS-orientated research project investigating how to build an exascale computer

- EESI is a European project working cloesely with IESP

Outstanding research priorities

- **systems software**
  - **operating systems:** fault-tolerance, collective OS services, power management, hierarchy management …
  - **runtime systems:** heterogeneity, load balancing, fault-tolerance, dynamical resource management …
  - **I/O systems:** integration of emerging storage devices, embed I/O into programming models …
  - **systems management:** resource control & scheduling, security, integration and test …
  - **external environments:** linking to remote resources …
- **development environments**
  - **programming models:** support for heterogeneous nodes, HPC interoperability, fault-tolerant MPI …
  - **frameworks:** data layouts, fault resilience, inter-component coupling …
  - **compilers:** MPI-aware compilers, compiler support for hybrid programming, power-aware compilers …
  - **numerical libraries:** asynchronous algorithms, architectural transparency, power-aware …
  - **debugging tools:** categorical assimilation, support for node heterogeneity, scalability …
- **applications**
  - **algorithms:** intra/inter-node scaling, fault resilience, heterogeneity, strong scaling …
  - **data analysis and visualisation:** integration with simulation, workflows, data extraction …
  - **data management:** scalable data-mining, new database technologies, search & query tools …
- **crosscutting activities**
  - **resilience:** techniques for saving/restoring state, MPI replacement, fault-oblivious software …
  - **power management:** node-level OS management, power-aware libraries etc …
  - **performance optimization:** heterogeneity, hybrid programming, enhanced concurrency …
  - **programmability:** new programming models, new runtime models, new compiler support …

IESP recommendations

- IESP strongly advocates a co-design model
  - The software and hardware are developed in parallel
- Backed by aspiration pull and technology push
  - Global challenges make the case … but the codes are too immature
  - Technology push is not enough
    - Politically the cost is too high, too few companies will benefit
    - Technically, there are many potential hardware routes … and many likely dead-ends
- Co-design vehicles
  - Applications which are scientifically sound with the potential to scale provide development paths
  - … while global challenge codes develop in parallel

Parallel computing today

- The programming model is one of a set distinct memories distributed over homogeneous microprocessors
  - Each microprocessor runs a Unix-like OS
- Data transfers between the processors are managed explicitly by the application
- Almost all programs are written in sequential Fortran or C
- They use MPI (Message Passing Interface) for data transfers between nodes/microprocessors
- Some applications which exploit parallel threads on each microprocessor use a hybrid model
  - Shared memory on the microprocessor, distributed memory outwith
  - This holds promise for many applications, but is still rare

Parallel computing today (cont)

- (Like the OS) few mathematical algorithms have been designed with parallelism in mind
  - … the parallelism is then “just a matter of implementation”
- This approach generates much duplication of effort as components are custom-built for each application
  - … but the years of development and debugging inhibits change and users are reluctant to risk a reduction in scientific output while rewriting takes place
- We may be close to a “tipping point”
  - Without fundamental algorithmic changes progress in many areas will be limited
- This doesn’t just apply to exascale
  - Some codes will soon fail to scale on an 8 or 16-core laptop

Bye bye homogeneity

- Today most HPC facilities are homogeneous
  - perhaps with specialised processors for peripheral functions, eg I/O
- Even if the nodes are compound, the components are separate with separate programming models
  - eg. microprocessors with attached FPGA
- Microprocessors will increasingly be built from disparate components: “normal” core, GPGPU, SIMD Array
  - With a mix which may vary within a machine
- … somehow, that mix will have to be controlled to give optimal performance

What is Europe doing in exascale?

- Initially funded a small project – EESI – to engage with IESP
  - Bringing all European players together
- European Commission have recently funded three exascale research projects – initial total funding of €25 million over three years
  - Projects start 1st October 2011 – currently negotiating contracts
- **CRESTA** – **C**ollaborative **R**esearch into **E**xascale **S**ystemware, **T**ools and **A**pplications
- 3 year project, 13 partners, €12 million costs, €8.5 million funding

KTH VETENSKAP OCH KONST

CREST

CRESTA

- CRESTA has a very strong focus on exascale software challenges
- Uses a co-design model of applications with exascale potential interacting with systemware and tools activities
- The hardware partner is Cray - following hardware trends closely
- Applications represent broad spectrum from science and engineering

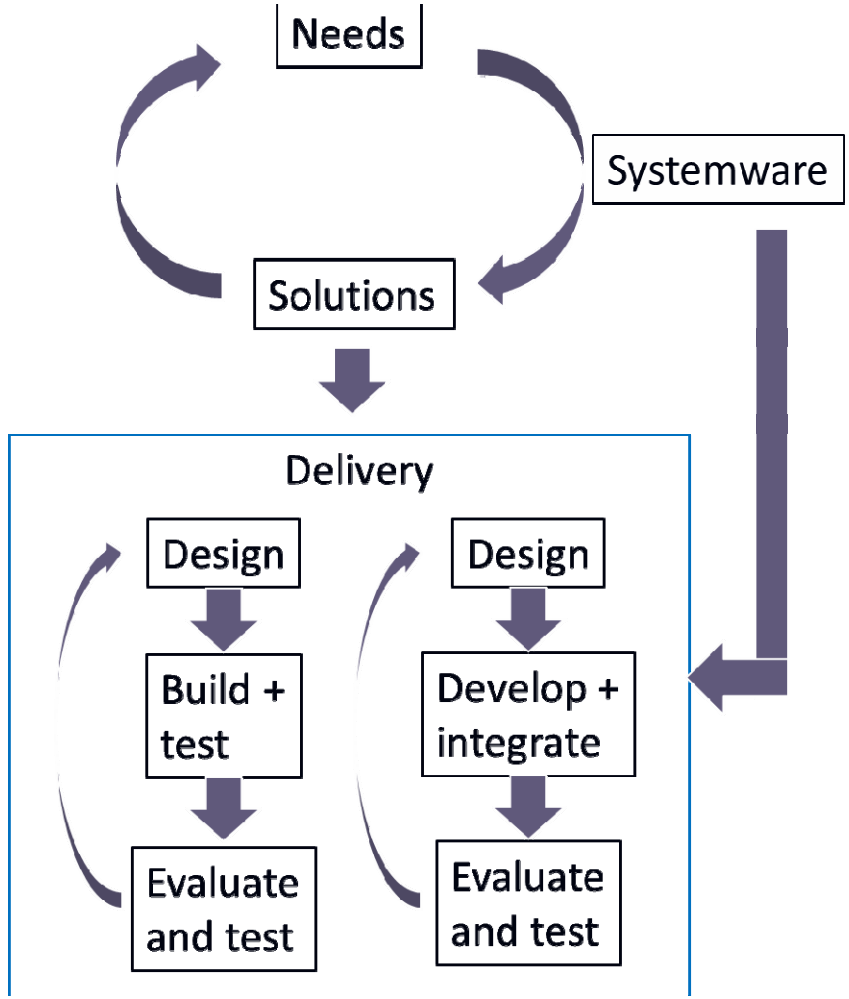
Application	Grand challenge	Partner responsible
GROMACS	Biomolecular systems	KTH (Sweden)
ELMFIRE	Fusion energy	ABO (Finland)
HemeLB	Virtual Physiological Human	UCL (UK)
IFS	Numerical weather prediction	ECMWF (International)
OpenFOAM	Engineering	EPCC / HLRS / ECP
Nek5000	Engineering	KTH (Sweden)

Partnership

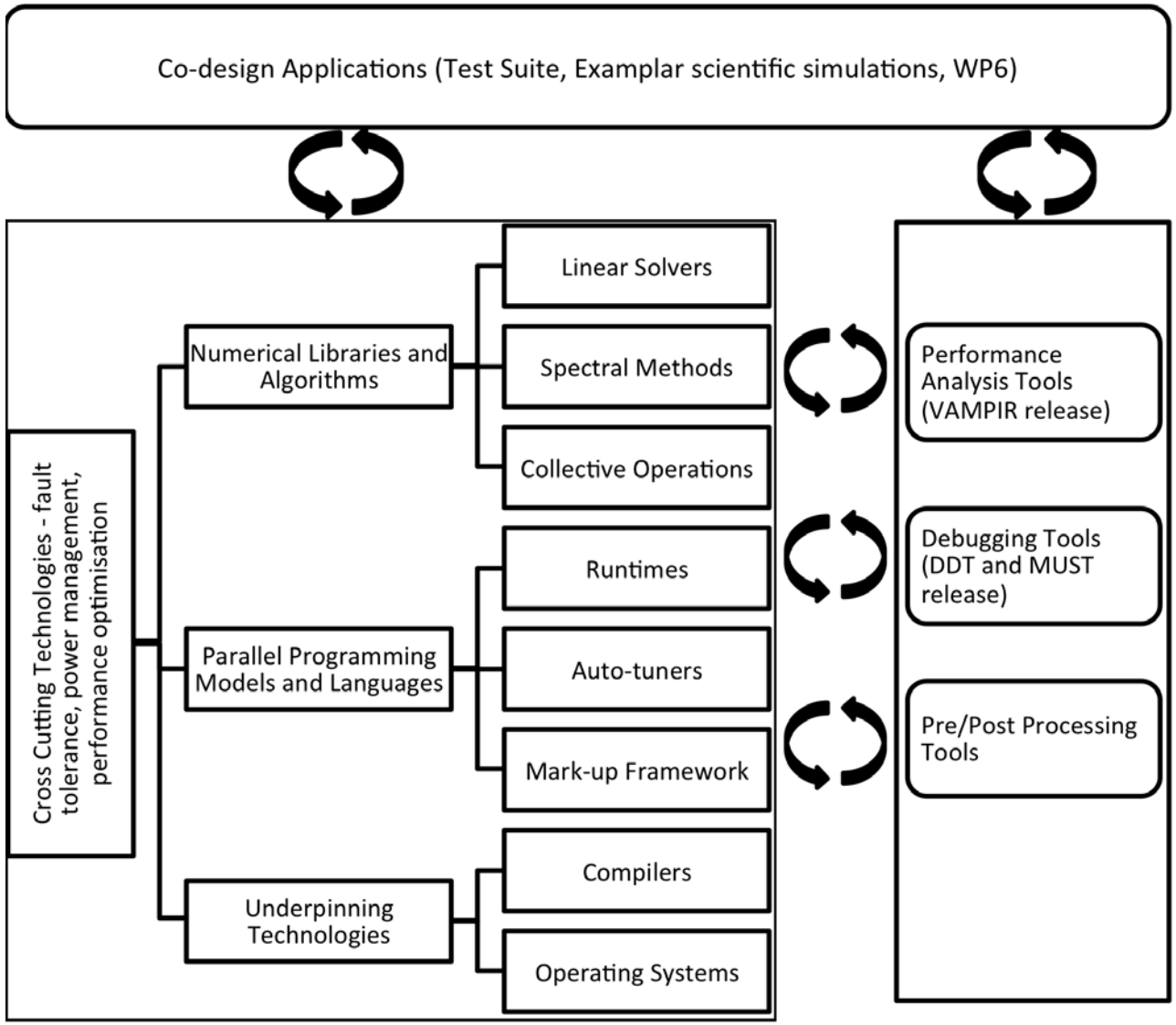
- Consortium has
  - Leading European HPC centres
    - EPCC, HLRS, CSC, PDC
  - A world leading vendor
    - Cray
  - World leading tools providers
    - TUD (Vampir), Allinea (DDT)
  - Exascale application owners and specialists
    - ABO, JYU, UCL, ECMWF, ECP, DLR, KTH
- CRESTA and its two partner projects are the first Exascale development projects funded by Europe

CRESTA key principle

- Some problems at exascale require incremental solutions
- Some problems at exascale require a disruptive approach
- CRESTA will compare and contrast different approaches
- This is particularly true for applications which are at the limit of scaling today
- What we learn at the exascale will help codes scale at the peta- and tera-scales.



CRESTA development model



Programming environments

Cresta provides support in all phases of the application lifecycle:

- Programming models that allow the construction of efficient, yet portable, applications
- Advanced compilation techniques and adaptive runtime environments
- Online and offline debugging
- Performance analysis

Programming Models

- Applications need to exploit every bit of parallelism:
  - Message passing (e.g. MPI)
  - Shared memory parallelism (e.g. OpenMP)
  - Vector and other instruction level parallelism
  - Accelerators (on-chip and over network)

- Cresta will explore hybrid programming models that allow a smooth evolution of the co-design codes.
  - Starting point will be MPI, OpenMP, PGAS

- Pragmatic approach: Markup framework
  - Identify and characterize parallelism in the application (programmer or automatically)
  - Include performance hints
  - Direct compiler and runtime systems
  - Feed back to OpenMP standardization

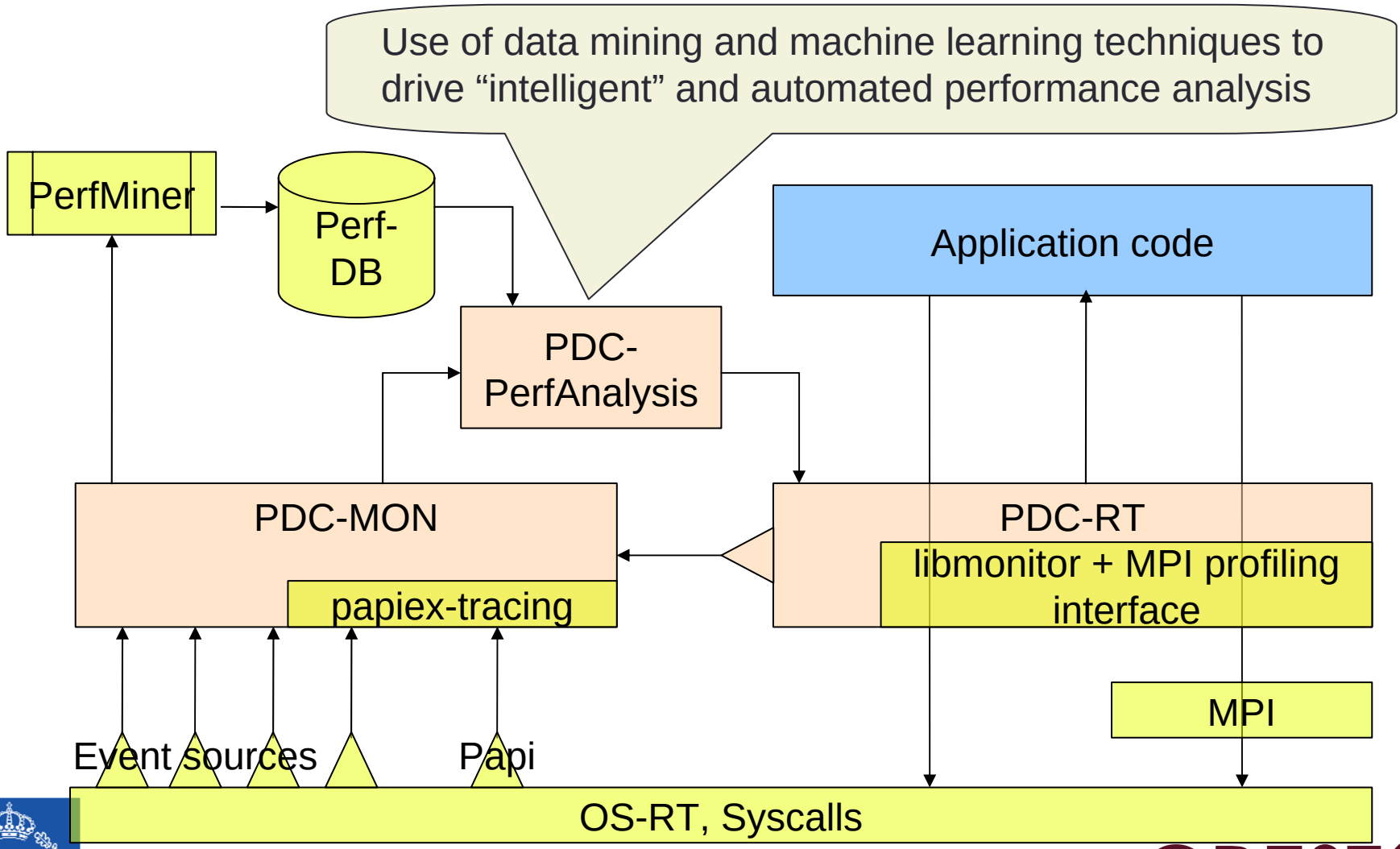
Compiler and runtime systems

- Auto-tuning
  - Develop a domain specific language for the expression of auto-tuning
  - Address code transformations, algorithmic choices, and runtime tuning
- Adaptive runtime system
  - Dynamically define or change task/hardware mapping
  - Support auto-tuning
  - Feedback loop from performance analysis tools
- Compiler framework
  - Assess different compiler frameworks
  - Integrate auto-tuning

Performance Analysis and Debugging

- Measure relevant performance data
  - Little emphasis so far on network performance
- Efficient data collection
  - Balance dynamically coarse- and fine-grained data collection
  - Manage potentially enormous amounts of data
- Analysis and visualization
  - Automatic analysis techniques to guide the user
  - Expert systems and machine learning techniques
- Debugging
  - Heterogeneous, dynamic debugging
  - Runtime error correction

Example: Dynamic feedback to adaptive runtime system



CRESTA objectives

- Main goal is to develop techniques and solutions which address the most difficult challenges that computing at the exascale can provide
- Success metrics by 2014
  - Co-design Applications
    - Co-design applications tested successfully on leading-edge petascale platforms and delivering previously unattainable simulations on those platforms
    - Roadmap to achieving application exploitation of exascale platforms
  - Systemware
    - Integrated CRESTA software stack successfully tested on petascale platforms
    - Co-design application simulations exploiting the CRESTA software stack to demonstrate massive and previously unattained scalability, reliability and usability
    - Roadmap to integrated CRESTA software stack on exascale platforms, reviewed by the Scientific Advisory Board

Conclusions

- We are at a fascinating point in supercomputing
  - The multi core revolution is posing complex questions
  - These questions span hardware, software and environmental issues
- But …
  - If we can build exascale systems the benefits could be enormous
  - We could simulate climate, molecules, vehicles in greater detail than ever before
  - We could use modelling and simulation to drive economic growth through better products and services
- If we're going to do this we need to invest more heavily than ever before in algorithms, software, systemware and hardware research
- GPGPUs and other accelerators are only part of the answer
- ***It's a highly parallel future!***