E-CAM Extreme-Scale State-of-the-art Workshop - 2017/07/06-07

POP CoE

`https://www.pop.coe.eu/`_ `mailto:pop@bsc.es`_

Multicore and memory revolution: * ISA/API leaks between hardware and applications * need to know sooo much to develop multicore apps * consequence: people focus more on computer technicalities than science * divergence the way machines behave and the way we think they behave * programmers need hope

Vision: rebuild an intermediate layer to decouple apps from hardware again. Switching focus from latency to throughput.

Framework: StarSs “granularities” * OmpSs * COMPSs / PyCOMPSs

POP CoE: * Performance Optmisiation and Productictivity * Promoting best practices in analysis of parallel performance

3 levels of services: * application performance audit: primary service (1 month) * application performance plan: follow-up / bottlenecks / suggested approaches (3 months) * proof-of-concept

Target customers: * users * developers * SMEs (explore sustainable business models)

Periodic tutorials

.. image: pop-perf-analysis.jpg

Behavior-oriented performance analysis (see picture): * flat profiles in terms of routines cause many global performance issues * factors describing serial behavior * factors modeling parallel efficiency * thresholds related to parallelization limitations ⇒ refactoring strategy * MPI is actually responsible of 25% of perceived MPI problems / main source = application itself * benchmarking only for big systems is like calculating the integral of a function between distant points ⇒ look at functional units varying parameters and sizes * variability diagram wrt many aspects (complexity, IPC, NUMAness, per iteration, clustering, …) * making “videos” of the runs and using techniques from security companies to track ill-behaved individuals

Refactoring: * OpenMP tasks + DLB * balancing IPC * better domain decomposition * to be customized wrt reach, skills, interest, …, of customer

Remarks: * If you go for a particular architecture, can you do it so that it is still valid for others? * It is better that vendors try to optimize their stuff than we spend time on it ourselves, even if it is far from being perfect.

Extreme-scale Demonstrators

EXDCI: European eXtreme DATA & Computing Initiative

.. dot:

```
digraph esds {
  HPC [label="HPC Ecosystem"];
  lobbies [label="Lobbying to EC\n(SRA, Work Proposals)"];
  ECWP [label="Work Programme"];
  calls [label="SRA Calls"];
  EC -> HPC;
  HPC -> ETP4HPC;
  ETP4HPC -> lobbies;
  lobbies -> EC;
  EC -> ECWP;
  ECWP -> calls;
  calls -> Projects;
  Projects -> HPC;
}
```

EU consumes 33%of HPC but provides < 5% and going down + many gaps in HPC landscape

EsDs concepts: * optimise and synergise whole EU HPC * tech providers: integration, system archtects, QA * application owners / CoEs: requirements and key challenges, porting, optimising * HPC Centers: co-design, deploy, validate, sysops * for EC: EsDs = milestone towards exascale * ETP: bring RIA results closer to commercialisation

Calls for proposals: * FETHPC WP14-15 * EsDs WP16-17 * 2 calls, 1st for pre-exascale * only RIA offers 100% funding rate and requires R&D * acquisition model: PCP, IPP

Remarks: * metrics should include scientific output, not only FLOPS

.. image: esds-eu-funding.jpg

The DEEP/-ER architecture

`http://www.deep-project.eu/`_

Hardware development: * General-purpose multicore processor technology * Single high-speed internal network * Problem: power consumption incompatible with exascale ⇒ accelerator * Flat topology of CPU+GPU nodes * Limitation: static assignment of accelerators to CPUs + lack of accelerator autonomy * Restructuring: cluster + booster connected by infiniband ⇒ flexibility * Prototype: 128 Xeon (cluster, low/medium scalability) + 384 Xeon Phi (booster, high scalability), 500 TFlops * Next generation: self-booting booster nodes, cache filesystem, on-node NVM, network-attached memory, simplified interconnect

Software development: * standard environment * open to any HPC code * OmpSs over MPI to ease the cluster → booster offload process * source code with pragmas (examples in C here) → OmpSs compiler → cluster executable + booster executabe * I/O: BeeGFS + extensions + SIONlib + Exascale10 * Resiliency software architecture: checkpointing, parallel analysis * applications: macroscopic, MD, high-energy physics

Strengths: * flexibility: dynamic use of booster, I/O offload * resource-efficient

Next step: modular computing architecture (see picture, DEEP-EST prototype)

.. image: deep-est-architecture.jpg

SHAPE

`http://www.prace-ri.eu/.../shape-programme/`_

EPCC: * 90 people * cycle sales and storage * training * consultancy and expertise * projects: directly funded; locally supported (SCS), EU funding * member of PRACE

SMEs: * 99% of businesses * 2015: 23 millions SMEs generated €3.9 trillion in value / 90 million people * EU potential: 142,000 SMEs could use HPC and 30,000 are likely to * HPC opportunities for SMEs: new products, optimize costs, < TTM * HPC barriers for SMEs: cost (€100k), expertise, resources ⇒ SHAPE

SHAPE: SME HPC Adoption Programme in Europe * supported by PRACE * raise awareness and assist SMEs wrt HPC * calls: every 6 months * 2-6 months of a PRACE expert * typical projects: 10-100 cores

Remarks: * interesting SMEs for EU projects are hard to find * the purpose is to improve what SMEs currently do, not reach extreme performance

PRACE Community Code Enablement

See PRACE 3IP, 4IP, and 5IP, documents

PRACE Advanced Training

From PRACE documents from 1IP to 5IP: * 6 centers * ≈ 1500 participants / year * PATC Programme for approval and organization of courses * Code Vault

CP2K

Specificities: * analytical internal representation Amn ↔ A(R) through grid colocation integration * possible Gaussian intermediates for representation * calculating functions of sparse matrices * small systems: dense linear algebra, large systems: Newton-Schultz iteration schemes for each matrix operation * Distributed Sparse Complex Raw Format: DBCSR library + index randomization for load balancing * Cannon's algorithm to reduce MPI communications * Small Matrix Multiplications (SMM): libsmm ⇒ 5 x faster than BLAS/LAPACK * Remote Memory Access (RMA): 2.5D multi-layer Cannon algorithm ⇒ weak scaling (almost constant up to 50kcores, < sqrt(n)) * applications: solvated nanoparticles (≈ 8000 atoms, 30ps), DFT calculation of a virus (8.4 million basis functions, a few fs) * wavefunction methods: Gaussian Plane Waves (GPW) + Resolution of Identity (RI) method, RPA, MP2, G0W0 * scaling wall ⇒ Reduced Scaling Methods (dRPA) ⇒ G0W0 with 1700 atoms * future: more

communication-avoidment algorithms

Scalability of Path-Sampling Simulations

Stochastic processes and rare events: * proper sampling way too long * one trajectory is not statistics, it's not even an anecdote * Flexible Length Shooting: much shorter trajectories, but MD monitoring necessary * OpenPathSampling: Python library (Larry Wall: “the 3 chief virtues of a programmer are laziness, impatience, and hybris”) * we are not computer scientists, we are computational scientists ⇒ how to answer a scientific question? * running trajectories in parallel whenever possible

Sparse matrices as an efficient tool in the extreme-scale calculations

---

Extreme-scale computing: * 1e18 ops / s * datasets of 1e17 bytes (e.g. digital sky survey) * the bigger the dataset, the higher probability to get near-zero elements ⇒ sparse matrices * trading speed for complexity * E-CAM module: DBCSR + MatrixSwitch, focus on usablity + documentation

PaPIM: A Code for (Quantum) Time Correlation Functions

Calculating time-dependent observables: * TD cross- and auto-correlation functions of observables * classical: polynomial scaling, quantum: exponential scaling * Phase Integration Method (PIM) ⇒ polynomial scaling * Trotter expansion + propagator linearisation * parallelisation of energy calculations * decoupling of classical trajectory propagation * automated parameter optimisation

Porting DL MESO DPD on GPUs

DPD: * similar to MD * coalescent memory access (adjacent cells) * DL_MESO_DPD: Fortran → arrays to C → copy to GPU * pros: energy-efficiency, speed, no need for a cluster * cons: 2 versions of the code, porting tedious

Optimised Reproducible Science

Provide code to industrial partners: * not only performance * predictability, reliability, usability, reproductibility * automatic deployment, verifiable

EasyBuild: * built from source * separate installations * highly optimised

Singularity: * containers for HPC * escape dependency hell * work the same anywhere * environment matters * same user in and out of container * Docker: would be almost OK, but not compliant with security policies

.. image: optimised-reproducible-science-containers.jpg

Performance metrics: * JUBE: configurble environment to run, monitor and analyse program workflows in a systematic way * JUBE used by EoCoE

MP2C

Particle-based hydrodynamics at scale: * solid-solvent systems * various methods * Multi-Particle Collision Dynamics:stochastic rotation dynamics, collision rules * MPC = local algorithm to describe collective properties * domain decomposition, no long-range interaction (for now) * random rotations to increase variety of collisions * solute / solvent separated (different time scales) and synchronized every n steps

Panel Discussion

Data locality is an essential criterion to decide how to set up the calculations.

During co-design, both ways from app to hardware and from hardware to app are explored at once to find the best match, within the limitations of each part.

The choice of an engine depends a lot on the system considered.

Use of Cassandra where the program tells which process treats which part of the data and the distribution among servers is done by Cassandra. Simutaneous run of Python analytics program.

NoMaD

Processing data all along its lifecycle

… Elaborated commercial prepared by a marketing professional …

Challenge of exascale: 1 million CPU hours / day producing exponential increase in data amounts

Interest of storing data: * repurposing results of previous calculations * use descriptors to get relevant information in a simpler way * access to regurlarly updated information

The user owns the data but 99% of the repository is open-access ⇒ like publications

EoCoE

Energy applications with one transversal WP to connect back to HPC, involving various people from various areas, with various languages ⇒ challenge: communicate efficiently

Exascale: * new schemes * improving software packages and integration * porting to some new architectures * performance evaluation and improvement on current applications * efforts driven by scientific objectives, not technical ones * data structures for petascale have to be redefined * integration instead of mini-apps

Expertise: * parallel sparse and hybrid solvers * algebraic multigrid preconditioners * focus: solver integration in applications * fault-tolerance, checkpointing * I/O: XIOS, SIONlib * PDI: Parallel Data Interface (describe kind of I/O and kind of toplogy to use)

Application performance: * multiple definitions * multiple references (what are we comparing to?) * not trivial to measure * 28 parameters automatically monitored

Performance Evaluation Workshops: * joint EoCoE-POP * 2-3 days * next: 2017/12/11-14 * devs come with test cases * done-with-you hands-on sessions * optimisation done by devs once back home * typically saves 40 million CPU hours by project after optimisation (ROI equals EoCoE expenses)

Open issues: * what will be the typical exascale architecture? * which applications will break through the petascale-exascale barrier?

Max —

Prepare applications for when EU sets up exascale architectures: * SIESTA, FLEUR, …, AiiDA * build knowledge on scientific problems requiring exascale * conduct proof-of-concept and field tests

Dennard scaling law for downscaling: * power gradient = square(density gradient) ⇒ power crisis * increasing cores to solve power crisis ⇒ programming crisis * CISC and RISC designed for maximum performance ⇒ predictible data center crisis (data overflow) * new chip design: maximum performance in a limited set of workloads ⇒ controlable throughput

Numbers: * global performance: 1e18 Flops * single FPU: 1e9 Flops max (transistor commutation physical limit) * exascale ≈ 1e4 servers * 1e5 FPUs but not all transistors can be powered at the same time

Predicitons: * exascale architectures will offer many options and selection procedures * will need to consider workloads at all stages * bring data in and out will be the main challenge

Activities: * stages: see image max-activities * libraries: see image max-libraries * preparing pre-exascale versions of various applications * communication avoidance techniques, double buffering, bindings with alternative MPI implementations * high-level API in Python

Remarks: * lot of work already done for QE

E-CAM Round Table

EU review aftermath: * what has been done so far? * how to match efforts and deliverables? * reinterpreting exascale and HPC in the context of E-CAM * the number of produced modules is not so important, as long as they come with a nice scaling plot

Mission, vision: * involve industrial partners beyond pilot projects * every single E-CAM member is part of a big team and should behave as such (no separate individualistic post-doc contracts) * no pilot project descriptions on website ⇒ contact companies one-by-one * highlight added value from E-CAM to all stakeholders

Methodology: * many ways to work despite unified tools (e.g. Gitlab) * bridge the gap between pilot projects and actual visible modules (WIP is fine) * move boring report contents to merge requests,

which can be discussed on-site * psychological barrier to restructure MRs in more modules during work * make MRs internal or private * sort out privacy issues for documentation on one side, and code on the other side * some information is required for management * WP leaders need to know what's going on * improve planning, workflows and monitoring to facilitate view of the big picture * how to build the correct narrative from the E-CAM as a whole perspective? * balance software infrastructure with scientific interest

Direct efforts to HPC: * high-throughput strategy towards extreme scale, common scheme? * pick one code to push it to extreme scale * co-design, adding features to codes: criticised by EU * developing modules AND use existing codes demonstrates capability of E-CAM to integrate communities * “intelligent farming” by providing the scientific community with a software infrastructure to organise their calculations * infrastructure: generic tool interfacing codes from different WPs (mappers), similar to AiiDa * bridging skill gaps: partnering with HPC Centers, PRACE, POP, … (e.g.: ESDW on Espresso++ ⇒ send people to PRACE courses on C++) * amend definition of ESDW: allow for transverse aspects (with partners) but require production of modules * ESDWs need implementers and drivers, with scientific motivation from the community

HPC resources to E-CAM post-docs: * preparatory: PRACE easy * estimates needed * can be batched

For each effort, identify and showcase: * industrial interest * what brings academy closer to industry * HPC contributions * relevant test cases (doesn't need to be the systems people want to study) * that the effort doesn't spoil the existing features

Off-talk discussions

About the E-CAM review: * impact of E-CAM modules and actions is measured by EU through Twitter * deliverables rejected ⇒ 20k€ withdrawn

About ESL Workshops: * pass on instructions from E-CAM to the participants * main issue: match actual work with what is visible within E-CAM

About exascale approaches: * POP: task-based programming * MAX: modularize and librarize incrementally * EoCoE: rewrite performance-critical parts from scratch