Agenda of the break-out session on Software

Yesterday

- Worked on tables listing software that are supposed to contribute to future Exascale stacks.
- Start from the tables of software provided by all regions for Exascale: Japan, Europe, USA, China, Russia
- Review of the components by Region and ask leaders to update their table
- Refined the definition (interregional instead of international)
- Long discussion about the classification
- Asked leader to update their table over the night with the new code and with mentioning already known collaborators

Today

- Improved (more details) tables adding collaborators and funding body
- → built a table that could be used as an instrument by funding agencies

Color Codes and Conventions

- In Column 1
  - Orange = Developers provide support
  - Green = Developers collaborate with vendor for vendor support
- Everywhere: Underline anything that is NOT open source
- Classes of Interregional Relations
  - C0 = Regional now but open to interregional
  - C1 = Cooperation (loose sharing, no interregional fund)
  - C2 = Collaboration (interregional funded joint effort)
  - C3 = Co-development with concrete software deliverable
  - C? = don't know at this point

Stack Components: Japan (updated April 13)

Group provides in production quality	Experimental versions only	No major research effort—rely on vendors or other countries
- Programming language *XScalableMP* (UTsukuba-Riken, Fujitsu) (C2, INRIA, JST-ANR), and accelerator extension XMP-dev (U. Tsukuba) (C2, INRIA, JST-ANR) - OS kernel for many core architecture MIC, node level scheduling, including PGAS style (UTokyo-Riken, Fujitsu, NEC, Hitachi, PCCC) (C0) - FT-MPI (C2, Riken, ORNL,JST) - File system-Lustre Enhancement (Fujitsu) (C3, OpenSFS* & WhamCloud*, K-funding) - File system-GFARM (UTsukuba) : (C2, INRIA, JST-ANR) - Communication libraries (below MPI layer) (UTokyo, Fujitsu, NEC, PCCC) (C0) - Scripting language for parallel workflow-Xcrypt (Kyoto U., Fujitsu) (C0) - *usa, with international partners	- FT frameworks & libraries-FTI, fault detectors(Tokyo Tech – UTokyo) (C2, INRIA-NCSA, JST-ANR; G8)(C1, LLNL) - Domain specific languages, *Physis* (Riken – TokyoTech) (C0) - Numerical libraries: z-Pares, Eigen-K (UTsukuba, AICS, etc.) (C0), - Autotuning functions (UTokyo, UTsukuba, KyotoU, etc.) (C2, INRIA-Saclay, ENSEEIHT-IRIT, Uversailles, JST-ANR) - "App Framework with Automatic Tuning" ppOpen-HPC (UTokyo, KyotoU, JAMSTEC) (C1, LBL) - "Application Framework" Famework (U-Tokyo)(C0) - Development Environment (Tohoku-U, U-Tsukuba, U-Tokyo) (C1, GWTH Aachen, UIUC, HLRS) - Communication library (space efficient, dynamic optim.) (KyushuU, Fujitsu, ISIT-Kyushu) (C0) - Power management and scheduling framework-scheduling strategies limiting peak power (Tokyo Tech.) (C2, NVIDIA, HP, GRC, etc., NVIDIA COE) - Data Analystics Framework (ChuoU-Tokyo Tech.) (C1)	Debuggers Performance tools Batch / Machine-wide scheduler Node level compilers for many-core

Stack Components: Europe (updated – April 13)

Group provides in production quality	Experimental versions only	No major research effort—rely on vendors or other countries
Programming models/ languages MPI/ OmpSs (C0), in the long run may move to OpenMP 4.0 (C3, multiple) Compiler for heterogeneous systems HMPP (C2, CAPS, ORNL, DOE) OpenACC (C3, multiple) Node-level runtime NANOS++ (C0), StarPU (C1, INRIA, UTK) (C2, Tsukuba, TiTech, JST-ANR) Performance Tools Paraver (C0), Scalasca (C1, Julich, UTK) (C2, JSC, Moscow, EU FP7) , Score-P (C3, multiple, Uoregon, DOE), Vampir (C2, TUD, ORNL, DOE), OPT (C1), ThreadSpotter (C1) Debuggers DDT (C2, Allinea, ANL, LLNL, ORNL, DOE) MPI Correctness MUST (C3, TUD,LLNL,DOE) Commit to contribute to MPI implementations MPICH, Open MPI (C1, multiple) MPI Madeleine (C2, INRIA, U. Tokyo, ANR-JST)	Communication libraries (below MPI) (C1, ?) , 1-sided communication library GPI (C1, ?) Fault tolerant libraries FTI, MPI (C1, INRIA, TiTech) Performance Tools Periscope (C0, TUM), Task-based Debugging Ayudame/Temanejo (C0, HLRS) Domain specific languages (for autotuning) (C?) Batch scheduler research MOAB (C1,ParTec, Cluster Resources) Power measurement management eeClust and others (C1, eeClust, ??) I/O middleware DAMARIS (C2, INRIA, UIUC, Joint Lab), SIONlib (C1, JSC, ???)	Node level compilers (vendor compilers) PGAS languages Scalable RAS system

Stack Components: Europe (updated – April 13)

Group provides in production quality	Experimental versions only	No major research effort—rely on vendors or other countries
File systems EOFS Lustre (C3, multiple, NEC, Netapp, Whamcloud) Numerical libraries (C1, multiple, UTK and others) Numerical libraries NAG (C1) OS kernel + cluster env BullX (C1) Log analysis HELO (C2, INRIA, UIUC, Joint Lab)	OAR batch scheduler (C0)	

Stack Components: USA (updated April 13)

Group provides in production quality	Experimental versions only	No major research effort—rely on vendors or other countries
Programming models: CAF *Fortran Standard Committee* (C2, multiple), OpenMP *Standard Committee* (C3, multiple) , OpenAcc *Consortium* (C3, multiple), OpenCL *Consortium* (C2, multiple) UPC *LBNL* (C0) OS Kernel *ANL SNL* (C1, ANL, U. Tokyo) FT Framework: Teal *IBM ANL LLNL NCSA* (C0) Integrated System Console *NCSA*(C2, INRIA, UIUC, Joint-lab) FT env: SCR (C1, *LLNL IBM, Titech*) Runtime: Charm++ *UIUC* (C0) Communication library: MPICH (C1, *ANL Cray IBM Intel Microsoft OSU UBC INRIA*) , Open MPI (C1, UTK, *ORNL Bull*) Tool infrastructure: MRNet (C0, *Cray UWI ORNL UNM*) Debugging: STAT (C0, *LLNL Cray*) TotalView (C0) Valgrind/Memcheck (C3, *OpenWorks, LLNL IBM, DOE*)	Performance tools: mpiP (C0), Fault tolerance backplane: CIFTS (C0, *ANL ORNL*) Compilers: Open64 (C1, *UHous ORNL NVIDIA, CAS, Tsinghua*) Compiler framework: Rose (C0) Compiler support for heterogeneous processing (C0, *UHous NVIDIA*) Gmac, dl, tc (C0, UIUC) Debugging and validation: MUST (C0) Runtime: Unistack/ Plum (C0) CCI (C1, INRIA, ANL, SNL, UTK) PowerMgmt Layer (C?) Low-level threading lib: Qthreads (C0) Task scheduler: ExM (C0) File system: PVFS (C0, multiple USA, UC3M, Hamburg), PLFS (C0) FT API (C0, *Cray UTAustin*) Data analytics libraries: Xanalytics (C0) Data model storage library: Damsel (C0) PnetCDF (C1, NorthW, ANL, U. Tokyo, RIKEN) Composition frameworks: COMPOSE-HPC (C0)	Commercial compilers

Stack Components: USA (updated April 13)

Group provides in production quality	Experimental versions only	No major research effort—rely on vendors or other countries
Performance tool: PAPI (C1, *UTK IBM*), OpenSpeedShop (C1, *Krell*), TAU (C1,, U. Or, *Juelich*), HPCToolkit (C0, *Rice*) Dyninst *(C1, U. Maryland, UWI, IBM, UAB, multiple)* File systems: Lustre (C3, *WhamCloud Cray LLNL ORNL, OpenSFS Xyratex, DDN, DOE and commercial*) I/O delegation (C0 Northwest., ANL) I/O middleware: IOFSL (C1, *ANL LANL ORNL SNL, U. Tokyo*) Visualization: VTK, Visit (C0) I/O libraries: HDF5, pNetCDF (C0/C1) Numerical libraries: MAGMA/ PLASMA (C1, INRIA, *KAUST, UCo* UCB), PETSc (C0/C1, *ANL*), Trilinos (C1, *SNL*), SuperLU (C1, UCB, INRIA), hypre (C0, LLNL) Resource managers: SLURM (C2/C3 *SchedMD LLNL Intel BULL HP Ubarcelona)*, MOAB. (C0, *Cluster Resources Cray IBM HP APPRO SGI etc*)	New programming models and approaches: Sketch, DSLs (C?), Chapel (C3, CRAY LLNL, SNL, BSC, U. Malaga), X10 (C2, *IBM, Titech, JST*) Benchmarks and mini-apps: SHOC (C0), SPP (C0, NCSA) DAGUE/QUARK –runtime DAG execution (C0)	

Stack Components: China (updated - April 13)

Group provides in production quality	Experimental versions only	No major research effort—rely on vendors or other countries
Operating system: Kylin OS (compatible with Linux, optimizations for HPC) (C0) File systems: Lustre enhancement (C1 NUDT, Whamcloud) Parallel Programming framework (structured mesh): JASMIN (C0) Light-weight fault-tolerant MPI: NR-MPI (C0) HPC virtual environment : HPVZ (C0) Resource management: Slurm enhancement (C1, NUDT, SchedMD)	Resilience computing framework (C2, NUDT, IIT Chicago, UIUC; NSFC) Hybrid tiered file system (C0) Heterogeneous programming runtime system (load balance & pipeline) (C0) Domain specific programming framework (unstructured mesh and others) (C0) Autonomic resource management and scheduler (C0) Integrated development tools (performance analysis, debugger, workflow) (C0)	Compilers for commercial processors

Stack Components: Russia

Group provides in production quality	Experimental versions only	No major research effort—rely on vendors or other countries
Numerical packages (FlowVision (TESIS) (C2), FireFly (C2), …) OS Kernel (C3) Software suit for multilevel programming of reconfigurable FPGA-computers (SFU) (C1) Programming language COLAMO (SFU) (C1) Batch scheduler Cleo (MSU) (C2) Batch scheduler SUPPZ (JSCC) (C1)	New programming models and languages (MSU, NUDA) (C1) Communication libraries (below MPI, DISLIB) (C1) Fault tolerant libraries (T-PI) (C2) Commit to contribute to MPI implementations (T-PI) (C?) Performance tools (MSU, LAPTA-HOPSA) (C3) Power measurement management (T-PI, RSC) (C?) I/O middleware (T-PI) (C?) Convergent cloud/HPC platform (RSC) (C1) Numerical libraries on GPU (SpVM): Sparse matrix-vector multiply, autotuning (ISP) (C1) Numerical packages: OpenFOAM on GPU (ISP) (C2) Compiler framework: automatic OpenCL generation in GCC (ISP) (C?) Tools and performance: Java+MPI bindings (ISP) (C?), HPC Virtualization: Palacios VMM improvements (ISP) (C1)	Debuggers, Compilers

Specific Areas Where Tight Collaboration would be useful

- Programming models (system level)
- Node level paradigms (manycores, accelerators, etc.)
- Performance and correctness tools
- Fault tolerance (system and application)
- Power efficiency
- Storage and file system
- Flexibility and adaptability

Specific Areas Where Tight Collaboration would be useful

- Observations:
  - Many P2P funding, almost no multi-region funding (>2)
  - No sufficient multi-region funding (>2) and no instrument
  - Very few US-to-Europe (and vice versa) funding compared to Japan-to-Europe (and vice versa)
- Planning to improve the situation
  - Consolidate the list along the topic area and make suggestions
  - Identify missing point: Comparison between the roadmap and the tables (gap analysis)
  - Providing Evidences of the benefit and success of these collaborations