Les Mardi du Développement Technologique
Towards a solver software stack on top of runtime systems

24th of November 2015
Outline

Context

Build solver software stack

Solutions and achievements

Next objectives
Outline

Context

Build solver software stack

Solutions and achievements

Next objectives
Matrices Over Runtime Systems @ Exascale

Linear algebra

\[ AX = B \]

Sequential-Task-Flow

for (j = 0; j < N; j++)
Task (A[j]);

Direct Acyclic Graph

Runtime systems

Heterogeneous platforms

A solver software stack on top of runtimes - Mardi du Développement Technologique - 24\textsuperscript{th} of November 2015
A view of HiePACS solvers

- **Chameleon**: dense linear solver
  - Tile algorithms: BLAS 3, some BLAS 2, LAPACK One-Sided, Norms
  - Supported runtimes: Quark and StarPU, (PaRSEC soon)
  - Ability to use cluster of heterogeneous nodes:
    - MPI+threads, CPUs (BLAS/LAPACK)+GPUs (cuBLAS/MAGMA)
A view of HiePACS solvers

PaStiX: sparse linear solver

- $LL^T$, $LDL^T$, and $LU$, with static pivoting, supernodal approach
- Native version: MPI+threads
- Versions with runtimes: on top of PaRSEC or StarPU
A view of HiePACS solvers

MaPHyS: hybrid direct/iterative sparse linear solver

- Solves $Ax = b$, where $A$ is a square non singular general matrix
- Native version: MPI+PaStiX/MUMPS+BLAS/LAPACK
- Do not support runtimes for now, work in progress
A view of HiPACS solvers

ScalFMM: scalable fast multipole methods

- Simulate N-body interactions using the Fast Multipole Method based on interpolation (Chebyshev or Lagrange)
- Native version: MPI+OpenMP+BLAS+FFTW
- Runtimes version: StarPU, OpenMP4 → StarPU (ADT K’STAR)
Solver on top of runtime system - example: Chameleon

**Chameleon**: dense linear algebra tile algorithms (STF) on top of runtime systems

1) Tile matrix layout

nb = 192, 320, 960, ... 

2) Algorithms

- STF Algorithms (**PLASMA**)
  - BLAS 3, some BLAS 2
  - LAPACK One-Sided, Norms
- Generic task submission interface
  - different runtimes can be plugged

3) Runtime systems

- QUARK: CPUs (pthread)
- StarPU: CPUs, CUDA, MPI

4) Optimized kernels

- BLAS, LAPACK
  - Intel MKL, Netlib blas/Lapack, ...
- CUDA/cuBLAS, MAGMA
- Misc: Hwloc, FxT, SimGrid, EZTrace, ...
How to deploy complex HPC software stacks?
Outline

Context

Build solver software stack

Solutions and achievements

Next objectives
Common situations

You rock: install all the stack by yourself !

- specialists on all the stack are rare
- problems of compatibility between versions
- takes a lot of time

You are a numerician not comfortable with software building

- ask someone else to do it
- use pre-installed versions (binary packages, modules)
  - problem: only a couple of versions exist
Wish list

Two kinds of users

1. Top-level users want the best version:
   • a default build with best options to get performances regarding the platform

2. A specialist wants to have the lead:
   • on the components he operates on
     ▶ flexibility to set his version
   • but may not care about many dependencies
     ▶ automatic choice of best options
Build solver software stack

Requirements

- A simple process to install a default version
- A flexible way to choose build variants
  - choose compiler, software versions
  - enable components, e.g. MPI: YES/NO
  - build options, e.g. --enable-debug
- Be able to install it on a remote cluster
  - no root permissions
  - no internet access (not necessarily)
Existing toolboxes

- **PETSc**
  - scientific library for solving PDEs in parallel MPI+Threads
  - wrappers to external solvers (partitionners, linear algebra, ...)
  - custom python scripts to activate packages
    - detection mode or download+install a web release, great!
    - detection problems, fixed versions to download

- **Trilinos**
  - similar to PETSc, maybe even broader scope
  - embed one precise version of solvers
  - no tool to install missing third party libraries
  
  ⇒ no competitive tool to install dependencies
Outline

Context

Build solver software stack

Solutions and achievements

Next objectives
Common build-install process for solvers

- use of CMake with similar options
- rely on the same detection of the system and libraries
  - recursive system of CMake Finds
  - if your application depends on Chameleon, in CMake:

```cpp
find_package(CHAMELEON COMPONENTS STARPU MPI CUDA MAGMA FXT)
```
Common build-install process for solvers

- use of CMake with similar options
- rely on the same detection of the system and libraries
  - recursive system of CMake Finds
  - if your application depends on **Chameleon**, in CMake:
    ```
    find_package(CHAMELEON COMPONENTS STARPU MPI CUDA MAGMA FXT)
    ```

List of available `find_package` in Morse:

<table>
<thead>
<tr>
<th>Category</th>
<th>Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>solvers</td>
<td>chameleon, magma, mumps, pastix, plasma, scalapack</td>
</tr>
<tr>
<td>runtimes</td>
<td>quark, parsec, starpu</td>
</tr>
<tr>
<td>kernels</td>
<td>(c)blas, lapack(e), fftw</td>
</tr>
<tr>
<td>misc</td>
<td>(par)metis, (pt)scotch, hwloc, fxt, eztrace</td>
</tr>
</tbody>
</table>

Available online:

https://scm.gforge.inria.fr/anonscm/svn/morse/trunk/morse_distrib/cmake_modules/morse/find
Solutions and achievements

State of the art: tool to distribute the software stack

- we do not want to reinvent the wheel i.e. use an existing solution:
  - Dpkg, 0install, Gub, Guix/Nix, Easybuild, ...
- classical package managers cannot meet our requirements
  - no root permissions, build variants easy to give, a mode to handle non open software (Intel MKL, nvidia CUDA)
- Spack a custom tool to install HPC libraries will be used
Solutions and achievements

**Spack**

http://scalability-llnl.github.io/spack/

- Python 2.7: no install needed, ready to be used

  $ git clone https://github.com/scalability-llnl/spack.git
  $ ./spack/bin/spack install gcc

- Easy way to set build variants, examples:

  $ spack install openmpi %gcc@4.9.2
  $ spack install netlib-lapack +shared
  $ spack install parpack ^netlib-lapack ^openmpi@1.10.0

- Handle modulefiles, mirrors to work on clusters

  $ spack load mpi
  $ spack mirror create openmpi mpich hwloc netlib-blas
  $ spack mirror add
Get Involved with Spack!

- **Come to our SC15 Talk:**
  
  *The Spack Package Manager: Bringing Order to HPC Software Chaos*

  Wednesday, 11:30am

  Hall 18AB

- Spack is starting to be used in production at LLNL
  - Build, test, and deployment by code teams.
  - Build research projects for students, postdocs.

- Spack has a rapidly growing external community.
  - NERSC is working with LLNL on Cray support for Cori.
  - Argonne/IIT cluster challenge project.
  - Kitware contributing ParaView builds & features.
  - Users at INRIA, EPFL, U. Oregon, Sandia, LANL, others.

Get Spack!


Spack weaknesses

- Not so mature $\Rightarrow$ bugs, not robust enough?
- Detection mode is missing ... but will be integrated soon
  - positive exchanges with the main developer
  - reactive to answer
  - we feel that they have the same needs
Morse in Spack: a fork where new packages can be found

- Available online - git repository:
  https://github.com/fpruvost/spack/ - morse branch

```bash
$ git clone https://github.com/fpruvost/spack.git
$ cd spack && git checkout morse
$ ./bin/spack install maphys
```

- Build variants examples:

```bash
$ spack install maphys ~examples +mumps
$ spack install pastix +starpu ^starpu@1.1.2 ^mkl-blas
$ spack install starpu@svn-1.2 +debug +cuda +mpi +fxt +examples
```

Tutorial online:

http://morse.gforge.inria.fr/tuto_spack-morse/tuto_spack.html
MORSE provides packages to automatically install libraries and its dependencies with Spack

- **Dense linear solvers**
  - Chameleon, MAGMA, PLASMA,
  - ScaLAPACK

- **Sparse linear solvers**
  - HIPS, MaPHyS, MUMPS, PaStiX,
  - qr_mumps, SuiteSparse

- **Fast Multipole Method solvers**
  - ScalFMM

- **Runtime systems**
  - QUARK, ParSEC, StarPU

- **Kernels**
  - (C)BLAS (MKL, Netlib, OpenBlas),
  - LAPACK(E), FFTW

- **Miscellaneous**
  - (Par)METIS, (PT-)SCOTCH, hwloc, MPI,
  - CUDA, SimGrid, EZTrace
Outline

Context

Build solver software stack

Solutions and achievements

Next objectives
Next objectives

Deliver a V0 of the solver stack distribution:

- Add some missing MPI packages
- Communication:
  - Documentation: prerequisites, features, limitations, compatibility issues
  - Website
Task-based solvers for users - an open question

Unify task-based solvers

- Installation of solvers is a step
  - common way to install the solver stacks
- What about their integration, usage in upper-level programs
- Is it possible to factorize something between solvers
  - Chameleon, Hips, MaPHyS, PaStiX, ScalFMM
    - change some solvers API, or
    - drive existing solvers with an intermediate layer (converters)
  - tutorial, documentation, vocabulary?
Thank you

ANY QUESTIONS?