R and High-Performance Computing

A (Somewhat Brief and Personal) Overview

Dirk Eddelbuettel

ISM HPCCON 2015 & ISM HPC on R Workshop

The Institute of Statistical Mathematics, Tokyo, Japan

October 9 - 12, 2015

Introduction

About Me

My background:

- Quant for about 20 years in Finance & Trading
- *Quants* predecessors to what is now *Data Science*
- Math Econ / Financial Econometrics Ph.D. + M.A.
- Industrial Engineering (Comp.Sci. / Op.Rsrch) M.Sc.
- As Quant, user of HPC for embarrassingly parallel tasks

About Me

My Open Source Work:

- Debian maintainer for R and more (& briefly Open MPI)
- R Project contributor, 25+ packages incl Rcpp & RInside
- R Foundation Board Member since 2014
- Co-author of survey paper on 'R and HPC' (JSS, 2009)
- Editor of CRAN Task View on HPC

Talk Today

Outline

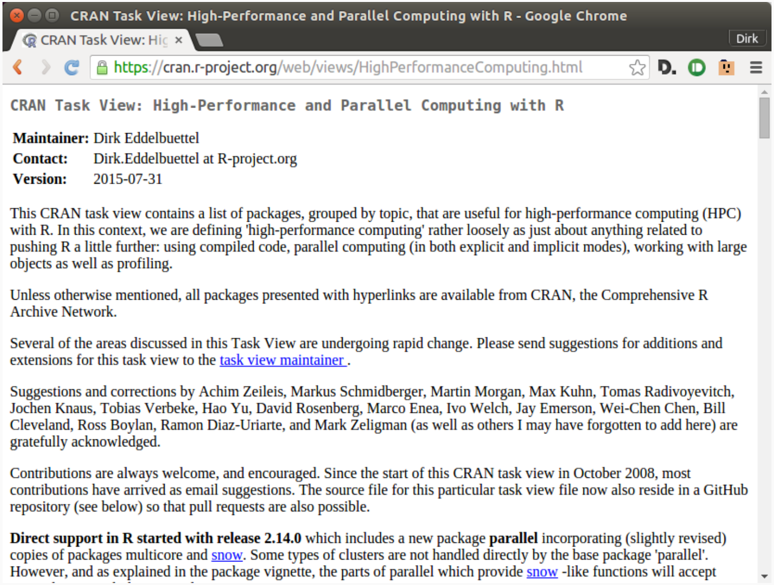
- (Brief) HPC Overview
    - "Bigger" Data
    - Parallel Computing
- Mostly personal views
- Somewhat pragmatic and applied

High-Performance Computing (HPC)

In a nutshell, HPC is working with

- (much) more data than single machine can handle
- (many) more processors than single machine has
- and some hybrid approaches tackle both.

High-Performance Computing (HPC)


CRAN Task View: High-Performance and Parallel Computing with R - Google Chrome
CRAN Task View: Hig
Dirk
https://cran.r-project.org/web/views/HighPerformanceComputing.html

CRAN Task View: High-Performance and Parallel Computing with R

Maintainer:	Dirk Eddelbuettel
Contact:	Dirk.Eddelbuettel at R-project.org
Version:	2015-07-31

This CRAN task view contains a list of packages, grouped by topic, that are useful for high-performance computing (HPC) with R. In this context, we are defining 'high-performance computing' rather loosely as just about anything related to pushing R a little further: using compiled code, parallel computing (in both explicit and implicit modes), working with large objects as well as profiling.

Unless otherwise mentioned, all packages presented with hyperlinks are available from CRAN, the Comprehensive R Archive Network.

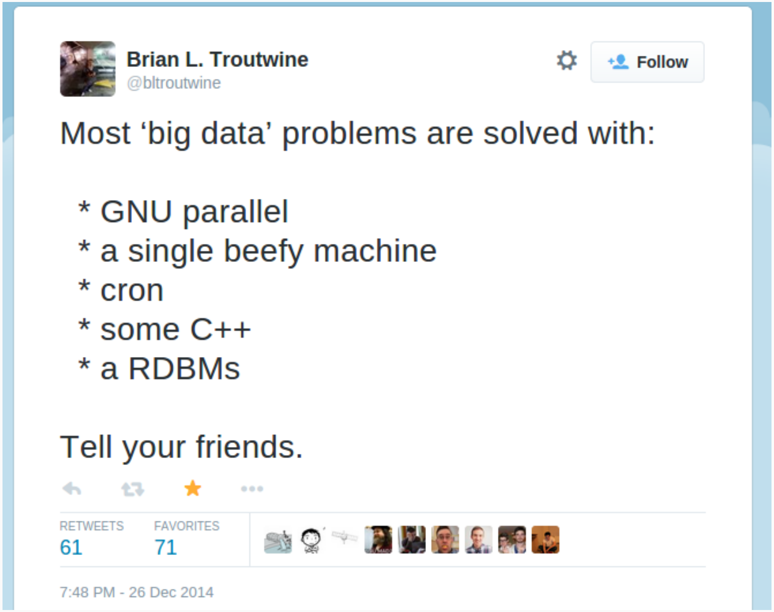
Several of the areas discussed in this Task View are undergoing rapid change. Please send suggestions for additions and extensions for this task view to the task view maintainer .

Suggestions and corrections by Achim Zeileis, Markus Schmidberger, Martin Morgan, Max Kuhn, Tomas Radivoyevitch, Jochen Knaus, Tobias Verbeke, Hao Yu, David Rosenberg, Marco Enea, Ivo Welch, Jay Emerson, Wei-Chen Chen, Bill Cleveland, Ross Boylan, Ramon Diaz-Uriarte, and Mark Zeligman (as well as others I may have forgotten to add here) are gratefully acknowledged.

Contributions are always welcome, and encouraged. Since the start of this CRAN task view in October 2008, most contributions have arrived as email suggestions. The source file for this particular task view file now also reside in a GitHub repository (see below) so that pull requests are also possible.

Direct support in R started with release 2.14.0 which includes a new package **parallel** incorporating (slightly revised) copies of packages multicore and snow. Some types of clusters are not handled directly by the base package 'parallel'. However, and as explained in the package vignette, the parts of parallel which provide snow -like functions will accept

Another Take



Brian L. Troutwine
@bltroutwine

Most ‘big data’ problems are solved with:

* GNU parallel
* a single beefy machine
* cron
* some C++
* a RDBMs

Tell your friends.

RETWEETS 61 FAVORITES 71

7:48 PM - 26 Dec 2014

Large Data

FF

CRAN Package FF

- Title: *“memory-efficient storage of large data on disk and fast access functions”*
- Description: *“The ff package provides data structures that are stored on disk but behave (almost) as if they were in RAM by transparently mapping only a section (pagesize) in main memory [...]”*
- Added Extra: Many more types representable than in R itself, potential savings in memory
- No development since March 2012

BigMemory

CRAN Package bigmemory

- Title: *"Manage massive matrices with shared memory and memory-mapped files"*
- Description: *"Create, store, access, and manipulate massive matrices. Matrices are allocated to shared memory and may use memory-mapped files. Packages biganalytics, bigtabulate, synchronicity, and bigalgebra provide advanced functionality."*
- Added Extra: Part of a suite of `big*` packages.
- Last update November 2013

PBDR

pdbR Project and CRAN Packages

- pdbR is a project out of Oak Ridge and U Tennesse
- It contains several packages including pdbBASE, pdbDMAT, pdbSLAP, pdbMPI, ...
- pdbDEMO is a good starting point and showcase
- See George Ostrouchov's talk at this workshop

Incremental Approaches

Some Problems Are Incremental

- The bigglm package can update `lm()` and `glm()` fits.
- Several packages emerging for streaming data processing
- E.g. Urbanek, Kane & Arnold on iotools, ioregression
- (Maybe?) See Ryan Hafen's talk at this workshop

Hybrid Approaches

Approaches Combining Large Data & Parallel Computing

- h2o reimplements several key R algorithms in Java; front-end h2o package on CRAN – see Erin LeDell's talk at this workshop
- HP/Vertica offers DistributedR with distributed data structures in C++, GPL'ed
- Dato (formerly GraphLab) recently released SFrame, another distributed data structure
- astoundingly enough, all three have their code on GitHub

Hybrid Approaches

And of course

- Hadoop as less exciting but commerically supported venue
- Spark is emerging as the next Hadoop

but I do not (currently) use either (though we remain curious about Spark)

New Approaches

dds: distributed data-structures

- Following a successful R/HPC focussed workshop in January, Roy (HP) and Lawrence (Genentech, R Core) have been working on a new API
- Distributed Data Structures (dds) aims to be a 'DBI'-alike layer on top of distributed data strucures
- Code still in private repo, release maybe one month out
- Per presentations at R Summit & useR! this summer, initial support for SparkR and Distributed R

And from the other side

Ibis

- Cloudera just released the Ibis project code on GitHub
- While aimed (primarily) at Python, it is C++-based and could provide an opportunity to form a common ‘large / distributed’ data structure backend
- Other Open Source projects could (in theory) deploy this
- In practice, this may never happen as there are *many* project-specific details

Parallel Computing

Parallel Computing for HPC

Overview

- Parallel Computing is still largely done via MPI
- MPI stands for *Message Passing Interface*: highly efficient communication
- Additional layers such as slurm add value to MPI
- Other approaches (cf the 'bigger data' section):
    - Spark
    - h2o
    - HP's distributed data structures
    - Dato's SFrame

MPI

```c
#include <stdio.h>
#include "mpi.h"

int main(int argc, char** argv) {
    int rank, size, nameLen;
    char processorName[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Get_processor_name(processorName, &nameLen);

    printf("Hello, rank %d, size %d on processor %s\n",
          rank, size, processorName);

    MPI_Finalize();
    return 0;
}
```

MPI

Running helloWorld for MPI

- Compile via `mpiCC helloMPI.c -o helloMPI`
- Run via `orterun [...] helloMPI`
- Arguments for nodes etc can be supplied.

helloWorld Output

```
$ orterun -n 10 /net/tmp/helloMPI
Hello, rank 2, size 10 on processor 141devfe03
Hello, rank 8, size 10 on processor 141devfe10
Hello, rank 0, size 10 on processor 141devfe01
Hello, rank 7, size 10 on processor 141devfe09
Hello, rank 1, size 10 on processor 141devfe02
Hello, rank 4, size 10 on processor 141devfe06
Hello, rank 5, size 10 on processor 141devfe07
Hello, rank 3, size 10 on processor 141devfe05
Hello, rank 9, size 10 on processor 141devfe01
Hello, rank 6, size 10 on processor 141devfe08
$
```

Rmpi

```r
#!/usr/bin/env r

library(Rmpi)   # calls MPI_Init

rk <- mpi.comm.rank(0)
sz <- mpi.comm.size(0)
name <- mpi.get.processor.name()
cat("Hello, rank", rk, "size", sz, "on", name, "\n")

mpi.quit()
```

helloWorld output

```
$ orterun -n 10 r /net/tmp/mpiHelloWorld.r
Hello, rank Hello, rank 4 size 10 on 141devfe06 5 size 10
on 141devfe07
Hello, rank Hello, rank 2Hello, rank  size 10 on 141devfe03
3 size 10 on 141devfe05
1 size 10 on 141devfe02
Hello, rank 8 size 10 on 141devfe10
Hello, rank 6Hello, rank  size 10 on 141devfe08
0 size 10 on 141devfe01
Hello, rank 9 size 10 on 141devfe01
Hello, rank 7 size 10 on 141devfe09
$
```

As a general rule, do not rely on `stdout` ...

Rmpi

Status

- very mature and standard package
- used by parallel (which is part of ‘base R’), snow, Rhpc, doMPI, pbdMPI, ...
- it is ‘fabric’ used for data parallel tasks

Rhpc

Rhpc as alternative to Rmpi / parallel

- faster / less overhead
- so scales better for larger number of nodes
- contains R serialization code (which I re-used in RApiSerialize package)
- related package RhpcBLASctl to control number of threads

pbdR MPI

pbdR MPI package

- As noted above, part of pbdR project
- no personal experience

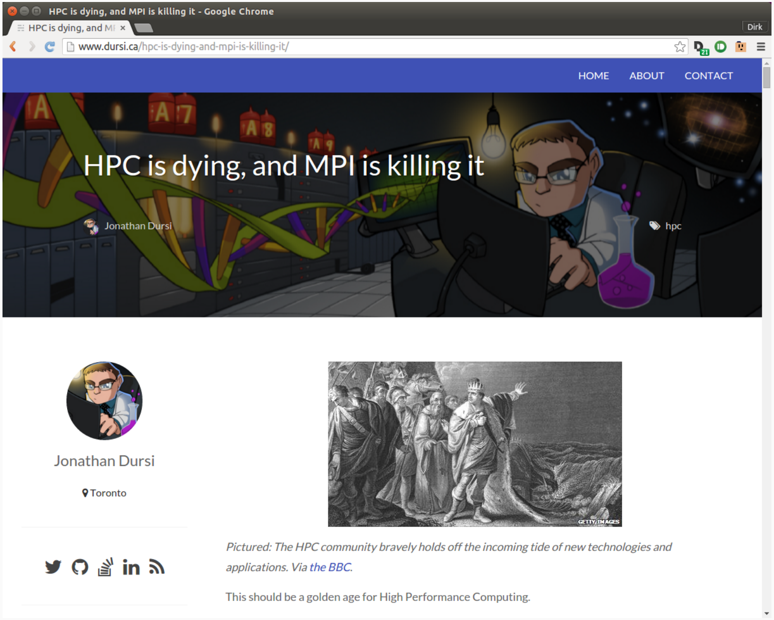
doMPI

doMPI package

- part of the foreach package set
- provides a *pluggable* backend
- permits each switching between serial modes, and doSNOW, doRedis, doMC, doParallel, ...
- that said, I always stuck with the snow package

Criticism

HPC is dying, and MPI is killing it



HPC is dying, and MPI is killing it - Google Chrome

www.dursi.ca/hpc-is-dying-and-mpi-is-killing-it/

HOME ABOUT CONTACT

HPC is dying, and MPI is killing it

Jonathan Dursi

hpc

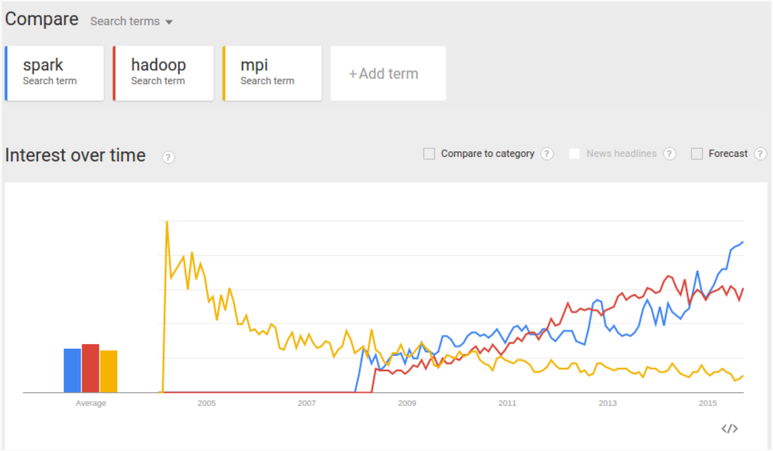
Jonathan Dursi

Toronto

Pictured: The HPC community bravely holds off the incoming tide of new technologies and applications. Via the BBC.

This should be a golden age for High Performance Computing.

(Updated) Hadoop vs Spark vs MPI Trends



With full credit to Jonathan Dursi's original post.

Big(ger) Data Case Study: Batch Processing

Slurm

Resource Manager for Clusters

- Heavy metal:

  Slurm provides workload management on many of the most powerful computers in the world. On the November 2013 Top500 list, five of the ten top systems use Slurm including the number one system. These five systems alone contain over 5.7 million cores.

- But also easy to use as it is part of Debian / Ubuntu too

Slurm at a glance

Commands

- `srun`, `salloc`, `sbatch` to launch
- `sinfo`, `scancel`, `squeue` to query
- `sview` simple GUI

Application at work

Simple Batch Processing

- Tie together a number of blades
- Add some dedicated larger systems
- Partitions *dev*, *prod*, *world*
- Typically simple data parallel runs
- Simple home-grown script on top of slurm
- Fancier R solutions provided by BatchJobs

A Complete Example

We launch the master *R* script via `sbatch`:

```
sbatch --array=1-125 --extra-node-info=2:1:1 \
       ./largeParameterExampleArray.r
```

By using the `--array` argument we can supply a range for an indexing variable (see below).

By using the `-B | --extra-node-info` argument, we can limit the resource utilization to two cores per node (much finer-grained control available, we currently allocate four core per compute node).

The controlling R script

This script constructs a grid of 5 x 5 x 5 parameters — and uses the *array* to index each call.

```r
#!/usr/bin/r

size  <- c("small", "medium", "large", "big", "huge")
speed <- c("immobile", "slow", "normal", "fast", "lightning")
color <- c("white", "yellow", "orange", "blue", "black")

runs <- expand.grid(size, speed, color)

slurmid  <- as.integer(Sys.getenv("SLURM_ARRAY_TASK_ID"))

pars <- runs[slurmid, ]

file <- sprintf("/net/tmp/job-array-%d.txt", slurmid)
cmd <- paste("./someBigModel.sh ", "--size", pars[1],
             "--speed", pars[2], "--color", pars[3], file)
system(cmd)
```

The controlling R script

The actual call is into a fake shell script (`someBigModel.sh`, see next slide) as we are just testing.

However, this scheme is *perfectly generic*:

- we span a grid of values
- set of parameters could come from file, db, ...,
- launch job with the array index we are called with,
- we can postprocess, report, summarize, ...

while slurm controls resource utilization, sharing, rights, ...

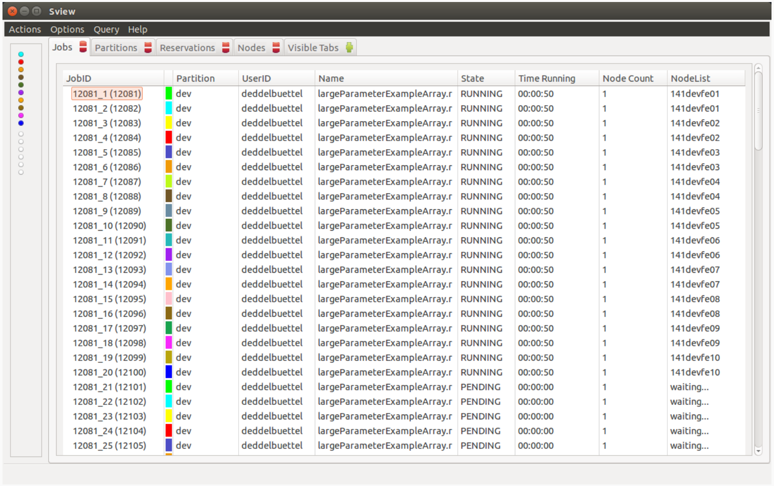
The (here fake) job script

This script just collects when it was called and finishes, and spawns a moderately expensive QuantLib example taking a few seconds (or minutes) to complete.

```bash
#!/bin/bash

echo -n $(date) ":"
#LatentModel > /dev/null
MarketModels > /dev/null
echo "Called with: $@" $(date) $(hostname)
```

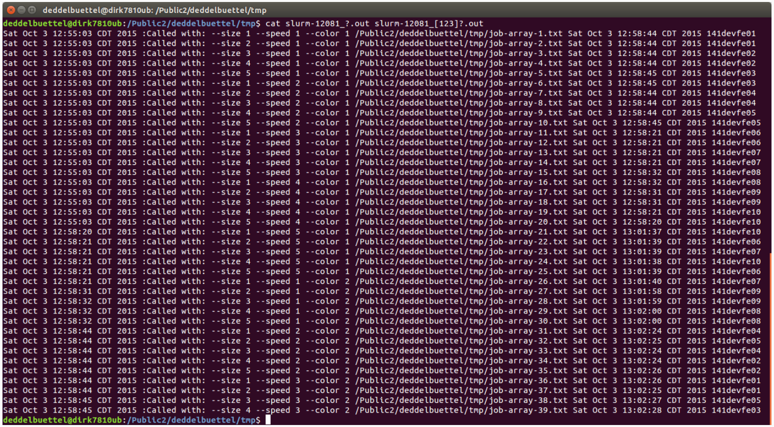
Screenshot



JobID		Partition	UserID	Name	State	Time Running	Node Count	NodeList
12081_1 (12081)	🟩	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe01
12081_2 (12082)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe01
12081_3 (12083)	🟨	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe02
12081_4 (12084)	🟥	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe02
12081_5 (12085)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe03
12081_6 (12086)	🟧	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe03
12081_7 (12087)	🟩	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe04
12081_8 (12088)	🟫	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe04
12081_9 (12089)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe05
12081_10 (12090)	🟩	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe05
12081_11 (12091)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe06
12081_12 (12092)	🟪	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe06
12081_13 (12093)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe07
12081_14 (12094)	🟧	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe07
12081_15 (12095)	🟪	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe08
12081_16 (12096)	🟫	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe08
12081_17 (12097)	🟩	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe09
12081_18 (12098)	🟪	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe09
12081_19 (12099)	🟨	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe10
12081_20 (12100)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	RUNNING	00:00:50	1	141devfe10
12081_21 (12101)	🟩	dev	deddelbuettel	largeParameterExampleArray.r	PENDING	00:00:00	1	waiting...
12081_22 (12102)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	PENDING	00:00:00	1	waiting...
12081_23 (12103)	🟨	dev	deddelbuettel	largeParameterExampleArray.r	PENDING	00:00:00	1	waiting...
12081_24 (12104)	🟥	dev	deddelbuettel	largeParameterExampleArray.r	PENDING	00:00:00	1	waiting...
12081_25 (12105)	🟦	dev	deddelbuettel	largeParameterExampleArray.r	PENDING	00:00:00	1	waiting...

Output

We can see the start and end times. Subsequent jobs were launched later as expected:



deddelbuettel@dirk7810ub: /Public2/deddelbuettel/tmp

```
deddelbuettel@dirk7810ub:/Public2/deddelbuettel/tmp$ cat slurm-12081_?.out slurm-12081_[123]?.out
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 1 --speed 1 --color 1 /Public2/deddelbuettel/tmp/job-array-1.txt Sat Oct 3 12:58:44 CDT 2015 141devfe01
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 2 --speed 1 --color 1 /Public2/deddelbuettel/tmp/job-array-2.txt Sat Oct 3 12:58:44 CDT 2015 141devfe01
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 3 --speed 1 --color 1 /Public2/deddelbuettel/tmp/job-array-3.txt Sat Oct 3 12:58:44 CDT 2015 141devfe02
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 4 --speed 1 --color 1 /Public2/deddelbuettel/tmp/job-array-4.txt Sat Oct 3 12:58:44 CDT 2015 141devfe02
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 5 --speed 1 --color 1 /Public2/deddelbuettel/tmp/job-array-5.txt Sat Oct 3 12:58:45 CDT 2015 141devfe03
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 1 --speed 2 --color 1 /Public2/deddelbuettel/tmp/job-array-6.txt Sat Oct 3 12:58:45 CDT 2015 141devfe03
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 2 --speed 2 --color 1 /Public2/deddelbuettel/tmp/job-array-7.txt Sat Oct 3 12:58:44 CDT 2015 141devfe04
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 3 --speed 2 --color 1 /Public2/deddelbuettel/tmp/job-array-8.txt Sat Oct 3 12:58:44 CDT 2015 141devfe04
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 4 --speed 2 --color 1 /Public2/deddelbuettel/tmp/job-array-9.txt Sat Oct 3 12:58:44 CDT 2015 141devfe05
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 5 --speed 2 --color 1 /Public2/deddelbuettel/tmp/job-array-10.txt Sat Oct 3 12:58:45 CDT 2015 141devfe05
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 1 --speed 3 --color 1 /Public2/deddelbuettel/tmp/job-array-11.txt Sat Oct 3 12:58:21 CDT 2015 141devfe06
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 2 --speed 3 --color 1 /Public2/deddelbuettel/tmp/job-array-12.txt Sat Oct 3 12:58:21 CDT 2015 141devfe06
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 3 --speed 3 --color 1 /Public2/deddelbuettel/tmp/job-array-13.txt Sat Oct 3 12:58:21 CDT 2015 141devfe07
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 4 --speed 3 --color 1 /Public2/deddelbuettel/tmp/job-array-14.txt Sat Oct 3 12:58:21 CDT 2015 141devfe07
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 5 --speed 3 --color 1 /Public2/deddelbuettel/tmp/job-array-15.txt Sat Oct 3 12:58:32 CDT 2015 141devfe08
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 1 --speed 4 --color 1 /Public2/deddelbuettel/tmp/job-array-16.txt Sat Oct 3 12:58:32 CDT 2015 141devfe08
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 2 --speed 4 --color 1 /Public2/deddelbuettel/tmp/job-array-17.txt Sat Oct 3 12:58:31 CDT 2015 141devfe09
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 3 --speed 4 --color 1 /Public2/deddelbuettel/tmp/job-array-18.txt Sat Oct 3 12:58:31 CDT 2015 141devfe09
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 4 --speed 4 --color 1 /Public2/deddelbuettel/tmp/job-array-19.txt Sat Oct 3 12:58:21 CDT 2015 141devfe10
Sat Oct 3 12:55:03 CDT 2015 :Called with: --size 5 --speed 4 --color 1 /Public2/deddelbuettel/tmp/job-array-20.txt Sat Oct 3 12:58:20 CDT 2015 141devfe10
Sat Oct 3 12:58:20 CDT 2015 :Called with: --size 1 --speed 5 --color 1 /Public2/deddelbuettel/tmp/job-array-21.txt Sat Oct 3 13:01:37 CDT 2015 141devfe10
Sat Oct 3 12:58:21 CDT 2015 :Called with: --size 2 --speed 5 --color 1 /Public2/deddelbuettel/tmp/job-array-22.txt Sat Oct 3 13:01:39 CDT 2015 141devfe06
Sat Oct 3 12:58:21 CDT 2015 :Called with: --size 3 --speed 5 --color 1 /Public2/deddelbuettel/tmp/job-array-23.txt Sat Oct 3 13:01:39 CDT 2015 141devfe07
Sat Oct 3 12:58:21 CDT 2015 :Called with: --size 4 --speed 5 --color 1 /Public2/deddelbuettel/tmp/job-array-24.txt Sat Oct 3 13:01:38 CDT 2015 141devfe10
Sat Oct 3 12:58:21 CDT 2015 :Called with: --size 5 --speed 5 --color 1 /Public2/deddelbuettel/tmp/job-array-25.txt Sat Oct 3 13:01:39 CDT 2015 141devfe06
Sat Oct 3 12:58:21 CDT 2015 :Called with: --size 1 --speed 1 --color 2 /Public2/deddelbuettel/tmp/job-array-26.txt Sat Oct 3 13:01:40 CDT 2015 141devfe07
Sat Oct 3 12:58:31 CDT 2015 :Called with: --size 2 --speed 1 --color 2 /Public2/deddelbuettel/tmp/job-array-27.txt Sat Oct 3 13:01:58 CDT 2015 141devfe09
Sat Oct 3 12:58:32 CDT 2015 :Called with: --size 3 --speed 1 --color 2 /Public2/deddelbuettel/tmp/job-array-28.txt Sat Oct 3 13:01:59 CDT 2015 141devfe09
Sat Oct 3 12:58:32 CDT 2015 :Called with: --size 4 --speed 1 --color 2 /Public2/deddelbuettel/tmp/job-array-29.txt Sat Oct 3 13:02:00 CDT 2015 141devfe08
Sat Oct 3 12:58:32 CDT 2015 :Called with: --size 5 --speed 1 --color 2 /Public2/deddelbuettel/tmp/job-array-30.txt Sat Oct 3 13:02:00 CDT 2015 141devfe08
Sat Oct 3 12:58:44 CDT 2015 :Called with: --size 1 --speed 2 --color 2 /Public2/deddelbuettel/tmp/job-array-31.txt Sat Oct 3 13:02:24 CDT 2015 141devfe04
Sat Oct 3 12:58:44 CDT 2015 :Called with: --size 2 --speed 2 --color 2 /Public2/deddelbuettel/tmp/job-array-32.txt Sat Oct 3 13:02:25 CDT 2015 141devfe05
Sat Oct 3 12:58:44 CDT 2015 :Called with: --size 3 --speed 2 --color 2 /Public2/deddelbuettel/tmp/job-array-33.txt Sat Oct 3 13:02:24 CDT 2015 141devfe04
Sat Oct 3 12:58:44 CDT 2015 :Called with: --size 4 --speed 2 --color 2 /Public2/deddelbuettel/tmp/job-array-34.txt Sat Oct 3 13:02:24 CDT 2015 141devfe02
Sat Oct 3 12:58:44 CDT 2015 :Called with: --size 5 --speed 2 --color 2 /Public2/deddelbuettel/tmp/job-array-35.txt Sat Oct 3 13:02:26 CDT 2015 141devfe02
Sat Oct 3 12:58:44 CDT 2015 :Called with: --size 1 --speed 3 --color 2 /Public2/deddelbuettel/tmp/job-array-36.txt Sat Oct 3 13:02:26 CDT 2015 141devfe01
Sat Oct 3 12:58:44 CDT 2015 :Called with: --size 2 --speed 3 --color 2 /Public2/deddelbuettel/tmp/job-array-37.txt Sat Oct 3 13:02:25 CDT 2015 141devfe01
Sat Oct 3 12:58:45 CDT 2015 :Called with: --size 3 --speed 3 --color 2 /Public2/deddelbuettel/tmp/job-array-38.txt Sat Oct 3 13:02:27 CDT 2015 141devfe05
Sat Oct 3 12:58:45 CDT 2015 :Called with: --size 4 --speed 3 --color 2 /Public2/deddelbuettel/tmp/job-array-39.txt Sat Oct 3 13:02:28 CDT 2015 141devfe03
deddelbuettel@dirk7810ub:/Public2/deddelbuettel/tmp$
```

Summary

Summing Up

In Brief

- HPC as a union of ‘much data’ and ‘much processing’
- Fashions change:
  - Hadoop and now Spark dominate discussions.
  - MPI is still here.
- For us, MPI still works well.
- Add-ons such as slurm add more value.

Colophon

Made using

- TeXlive 20141024
- Beamer with mtheme
- Pandoc 1.12.4.2
- R 3.2.2
- rmarkdown 0.7
- Emacs 24.4
- ESS 15.09
- Ubuntu 15.04