Frequently and Not-So-Frequently Asked PETSc-Questions

Karl Rupp

me@karlrupp.net

Freelance Computational Scientist
and
Institute for Microelectronics, TU Wien

PETSc User Meeting 2017

Boulder, Colorado

June 14-16, 2017

Table of Contents

What about threads in PETSc?

What about all those branches in the repository?

What kind of questions do users have?

What is Firetran?

What about threads in PETSc?

PETSc and Threads

Competing Threading Approaches

- pthread
- OpenMP
- C++11 threads
- Compiler magic

Issues with Threads

- Problematic across compilers
- Data locality
- Thread ownership
- Software interface

Assumption for Subsequent Discussion

- Primary Goal: Get the science done!
- 10 percent performance difference is **not significant**

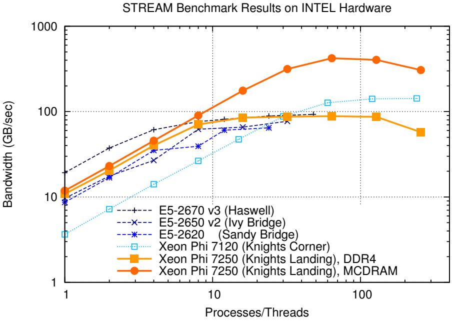
PETSc and Threads

When to Use Threads After All?

Distributed Memory: Need MPI anyway

Shared Memory: MPI for multi-socket systems for NUMA reasons

1-Socket Machines: If a 2-5x gain is critical, use a cluster!



Threads and Library Interfaces

Attempt 1

Library spawns threads

```
void library_func(double *x, int N) {
  #pragma omp parallel for
  for (int i=0; i<N; ++i) x[i] = something_complicated();
}
```

Problems

Call from multi-threaded environment?

```
void user_func(double **y, int N) {
  #pragma omp parallel for
  for (int j=0; j<M; ++j) library_func(y[j], N);
}
```

Incompatible OpenMP runtimes (e.g. GCC vs. ICC)

Threads and Library Interfaces

Attempt 2

Use pthreads/TBB/etc. instead of OpenMP to spawn threads

Fixes incompatible OpenMP implementations (probably)

Problems

Still a problem with multi-threaded user environments

```
void user_func(double **y, int N) {
  #pragma omp parallel for
  for (int j=0; j<M; ++j) library_func(y[j], N);
}
```

Threads and Library Interfaces

Attempt 3

Hand back thread management to user

```
void library_func(ThreadInfo ti, double *x, int N) {
  int start = compute_start_index(ti, N);
  int stop  = compute_stop_index(ti, N);
  for (int i=start; i<stop; ++i)
    x[i] = something_complicated();
}
```

Implications

Users can use their favorite threading model

API requires one extra parameter

Extra boilerplate code required in user code

Threads and Library Interfaces

Reflection

Extra thread communication parameter

```
void library_func(ThreadInfo ti, double *x, int N) {...}
```

Rename thread management parameter

```
void library_func(Thread_Comm c, double *x, int N) {...}
```

Compare:

```
void library_func(MPI_Comm comm, double *x, int N) {...}
```

Conclusion

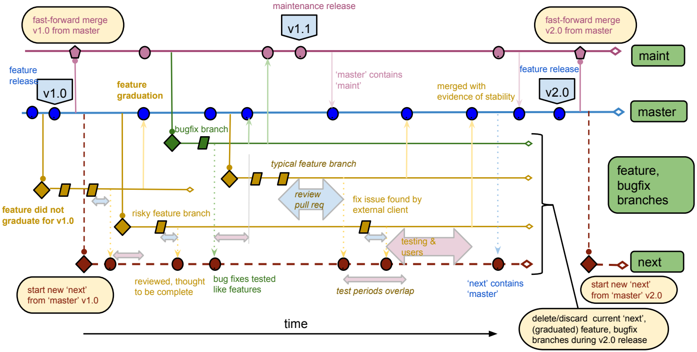
Prefer flat MPI over MPI+OpenMP for a composable software stack

MPI automatically brings better data locality

What about all those branches in the repository?

PETSc and Git

PETSc's Workflow


maintenance release
v1.1
fast-forward merge v1.0 from master
fast-forward merge v2.0 from master
maint
feature release
v1.0
feature graduation
'master' contains 'maint'
merged with evidence of stability
feature release
v2.0
master
bugfix branch
typical feature branch
review pull req
fix issue found by external client
feature, bugfix branches
feature did not graduate for v1.0
risky feature branch
testing & users
next
start new 'next' from 'master' v1.0
reviewed, thought to be complete
bug fixes tested like features
test periods overlap
'next' contains 'master'
start new 'next' from 'master' v2.0
delete/discard current 'next', (graduated) feature, bugfix branches during v2.0 release
time

🟪 maint branch (stable, bugfix, release)

🟦 master branch (stable, new features, devel)

🟥 next branch (testing)

🟧 feature branch - usually starts from master

🟩 bugfix branch - usually starts from maint

—◇ a git branch

—◆ create a new git branch from the indicated branch/commit

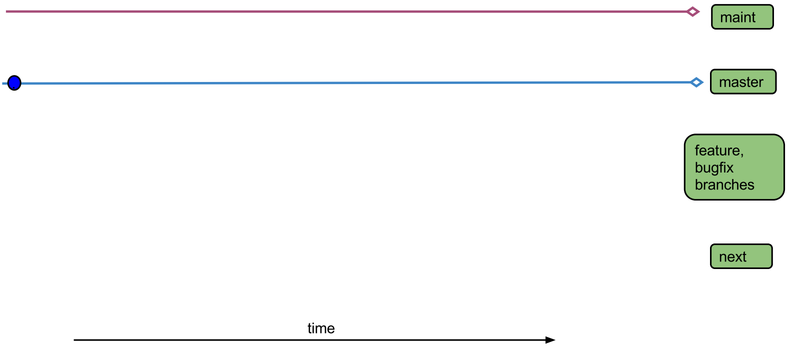
—▰ create a regular commit (with one parent) in current branch

—● create a merge commit. Dark line is from first parent (current branch). Light (arrow) line is from feature/bugfix/other branch

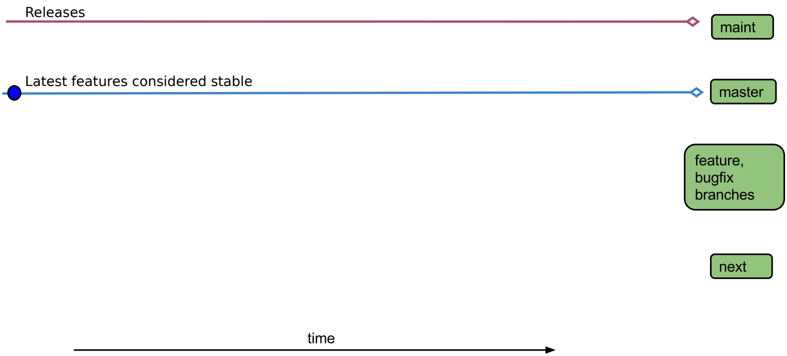
—⬟ merge "release" from master to maint (usually fast-forward)

- - - - this git history is discarded at the next release

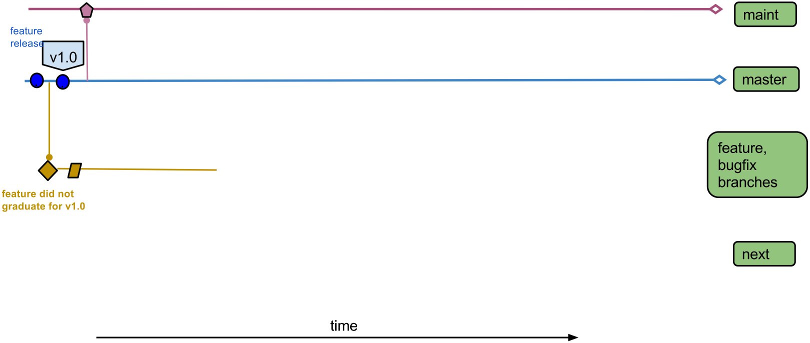
PETSc and Git



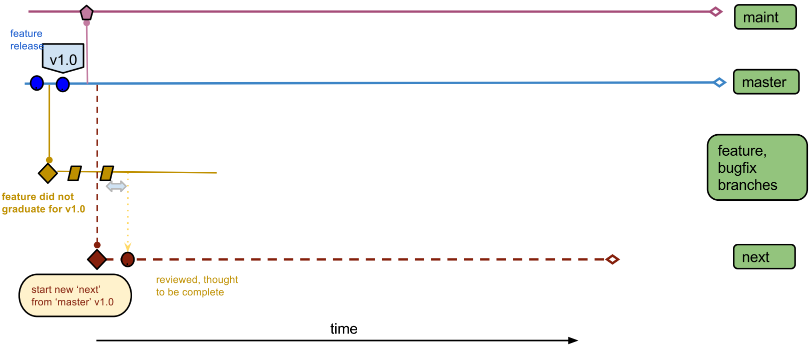
PETSc and Git



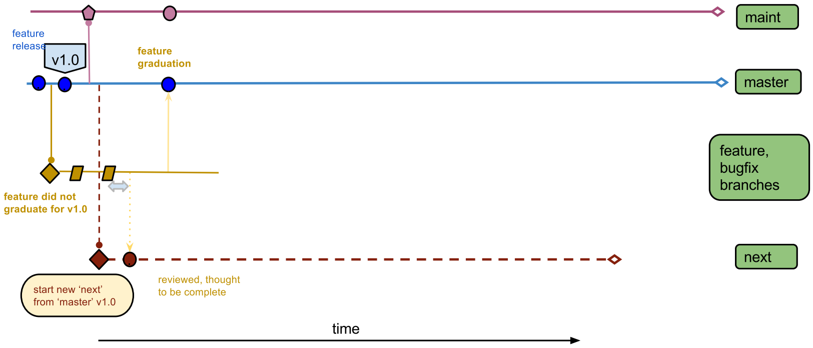
PETSc and Git



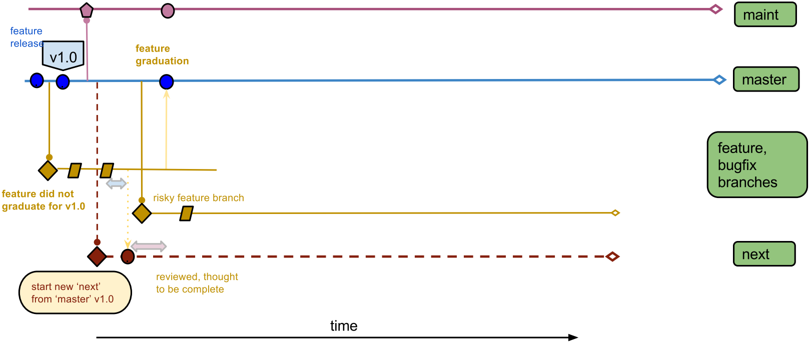
PETSc and Git


maint
feature release
v1.0
master
feature, bugfix branches
feature did not graduate for v1.0
next
start new ‘next’ from ‘master’ v1.0
reviewed, thought to be complete
time

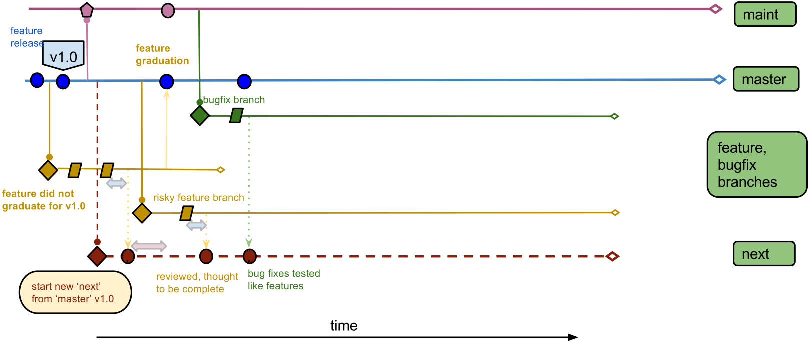
PETSc and Git


maint
feature release
v1.0
feature graduation
master
feature, bugfix branches
feature did not graduate for v1.0
next
start new ‘next’ from ‘master’ v1.0
reviewed, thought to be complete
time

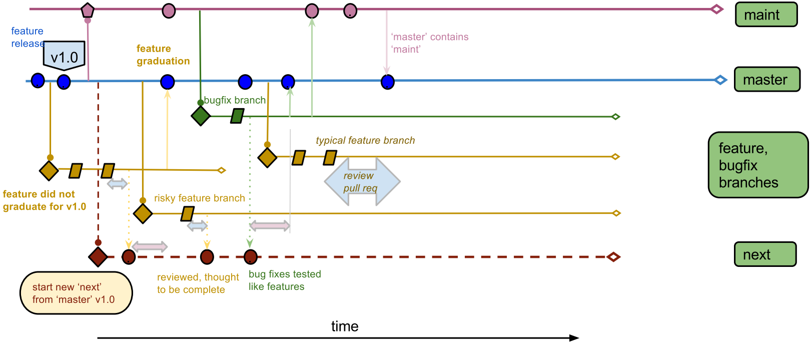
PETSc and Git


maint
feature release
v1.0
feature graduation
master
feature, bugfix branches
feature did not graduate for v1.0
risky feature branch
next
start new ‘next’ from ‘master’ v1.0
reviewed, thought to be complete
time

PETSc and Git

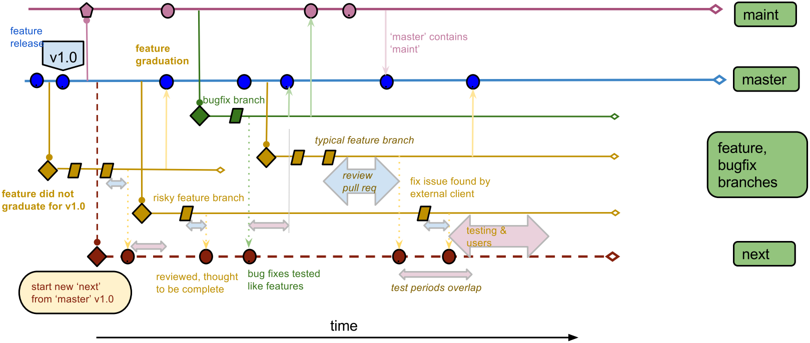

maint
master
feature, bugfix branches
next
feature release
v1.0
feature graduation
bugfix branch
feature did not graduate for v1.0
risky feature branch
start new 'next' from 'master' v1.0
reviewed, thought to be complete
bug fixes tested like features
time

PETSc and Git



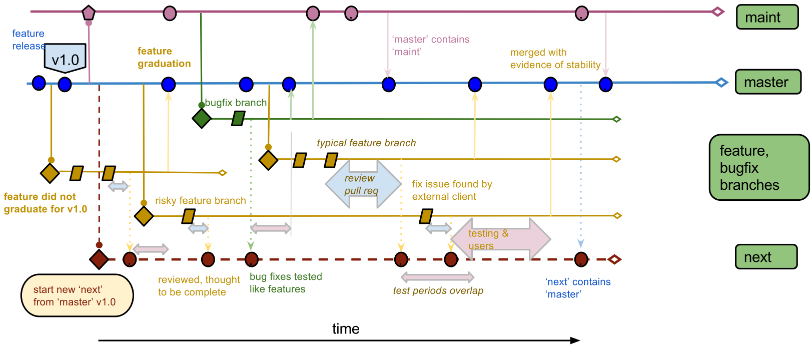
maint
master
feature, bugfix branches
next
feature release
v1.0
feature graduation
'master' contains 'maint'
bugfix branch
typical feature branch
review pull req
feature did not graduate for v1.0
risky feature branch
start new 'next' from 'master' v1.0
reviewed, thought to be complete
bug fixes tested like features
time

PETSc and Git

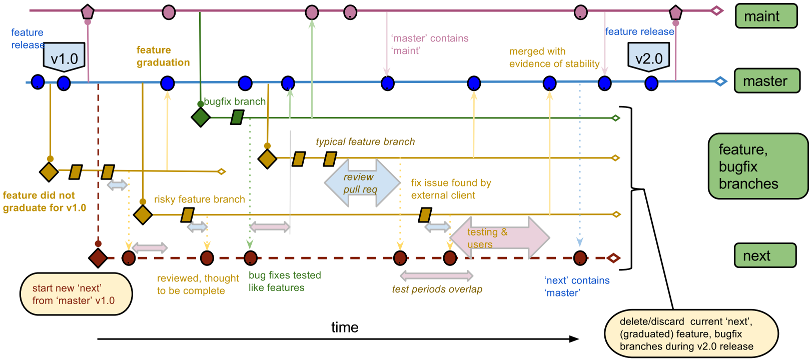


maint
master
feature, bugfix branches
next
feature release
v1.0
feature graduation
'master' contains 'maint'
bugfix branch
typical feature branch
review pull req
fix issue found by external client
feature did not graduate for v1.0
risky feature branch
testing & users
start new 'next' from 'master' v1.0
reviewed, thought to be complete
bug fixes tested like features
test periods overlap
time

PETSc and Git


maint
master
feature, bugfix branches
next
feature release
v1.0
feature graduation
bugfix branch
'master' contains 'maint'
merged with evidence of stability
typical feature branch
review pull req
fix issue found by external client
feature did not graduate for v1.0
risky feature branch
testing & users
start new 'next' from 'master' v1.0
reviewed, thought to be complete
bug fixes tested like features
test periods overlap
'next' contains 'master'
time

PETSc and Git



maint

master

feature, bugfix branches

next

feature release

v1.0

feature graduation

'master' contains 'maint'

merged with evidence of stability

feature release

v2.0

bugfix branch

typical feature branch

review pull req

fix issue found by external client

feature did not graduate for v1.0

risky feature branch

testing & users

start new 'next' from 'master' v1.0

reviewed, thought to be complete

bug fixes tested like features

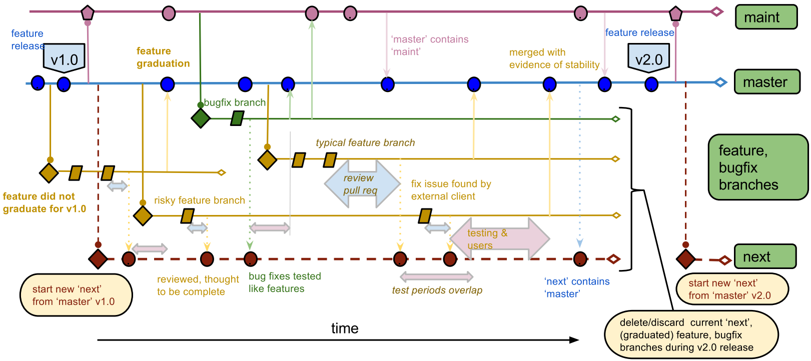
test periods overlap

'next' contains 'master'

time

delete/discard current 'next', (graduated) feature, bugfix branches during v2.0 release

PETSc and Git



maint

master

feature, bugfix branches

next

feature release

v1.0

feature graduation

bugfix branch

'master' contains 'maint'

merged with evidence of stability

feature release

v2.0

typical feature branch

review pull req

fix issue found by external client

feature did not graduate for v1.0

risky feature branch

testing & users

start new 'next' from 'master' v1.0

reviewed, thought to be complete

bug fixes tested like features

test periods overlap

'next' contains 'master'

start new 'next' from 'master' v2.0

delete/discard current 'next', (graduated) feature, bugfix branches during v2.0 release

time

What kind of questions do users have?

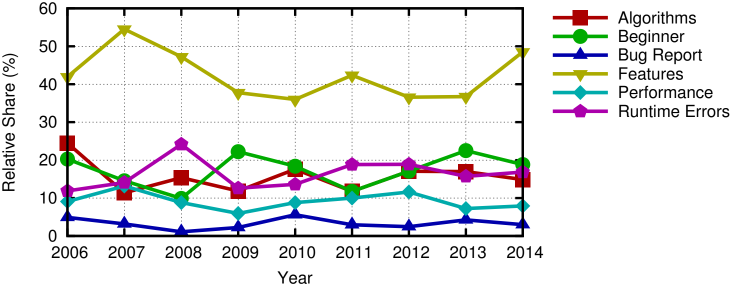
PETSc and Email Support

Study Setup

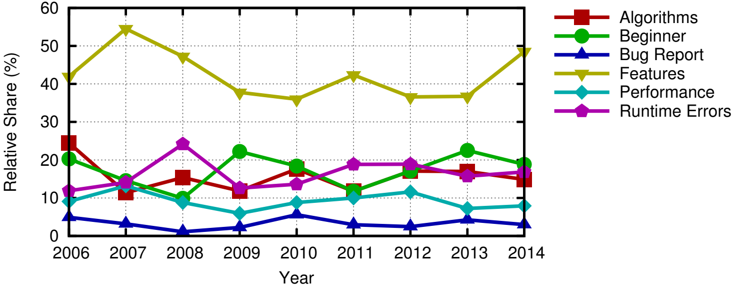
Classification of Email threads for petsc-user

Years 2006 - 2014

Volume: 904 emails in 2006, 3947 emails in 2014



PETSc and Email Support



Conclusion

Involvement of core developers essential

No significant shift in type of support emails

“Better” documentation would not change email volume significantly

What is Firetran?

My New Web Browser

after J. Brown, M. Knepley, B. Smith: “Run-time extensibility and librarization of simulation software” (2015)

Firetran Features

Performance

10 percent faster than Firefox ...

... but no JavaScript

Recompile for JavaScript, performance not shown

Character encoding compiled in for efficiency

Plugins

Directly coded in the core, guarded by `#ifdef`

Some ship their own version of Firetran

Code Quality

Some send bug reports

Ignore if not reproducible with my version

Firetran Features

Convenience

Proxy configuration compiled in, don’t worry about runtime dialogs

To change: Just edit makefile and recompile

Security

Firetran speaks either http or https

Firetran-http and Firetran-https

Open Source

Development done in private

Release when paper comes out

“We fixed that in our private repository last year”

Fill out attached form and fax signed copy to view our repository

Firetran Features

Parental Filter

up to 16 pages compiled in

Building Firetran

Build with last year’s version of ACME Fortran 77

Build system consists of csh, perl, m4, BSD make

Navigation

No URL bar to save precious screen space

URL entry in configuration file, restart Firetran

Student wrote Tcl script to automate editing and restarting

Script hard to understand, but the way forward

Firetran Problems

Only One Problem

Firetran struggles with market share

Reasons

Laughably unacceptable for non-scientific software

Firetran Symptoms?

Some Linear Algebra and Solver Libraries for Many-Core

MAGMA, clMAGMA, MAGMA MIC: CUDA, OpenCL, OpenMP/AVX

MKL: Web tool for linker flags

SuperLU, SuperLU_dist: Shared vs. distributed memory

Paralution, VexCL: Backend selection at compiletime

ViennaCL: Enable backends at compiletime

(long list to continue...)

Possible Approaches to Fix

Raise awareness for usability issues

Incentives for better software engineering

Don’t penalize scientists for writing reusable software

Conclusions

PETSc can Help You

- Solve algebraic and DAE problems in your application area
- Rapidly develop efficient parallel code, can start from examples
- Develop new solution methods and data structures
- Debug and analyze performance
- Advice on software design, solution algorithms, and performance

`petsc-{users,dev,maint}@mcs.anl.gov`

You can Help PETSc

- Report bugs and inconsistencies, or if you think there is a better way
- Tell us if the documentation is inconsistent or unclear
- Consider developing new algebraic methods as plugins, contribute if your idea works

Thank you!