Commit Quality in Five High Performance Computing Projects

Kapil Agrawal, Sadika Amreen, and Audris Mockus

University of Tennessee, Knoxville

{*kagrawa1@vols,samreen@vols,audris@*}*utk.edu*

International Workshop on Software Engineering for High Performance Computing in Science, Firenze, Italy May 27, 2015

Outline

1. Motivation and Goals
2. Context
3. Method
   - Approach
   - Measures
4. Results
   - Fraction of Unique Commit Comments
     - Discussion
   - Number of Delta and Comment Length
     - Discussion
5. Conclusion
6. Questions

Motivation and Goals

What are software development practices in High Performance Computing (HPC)?

Objective

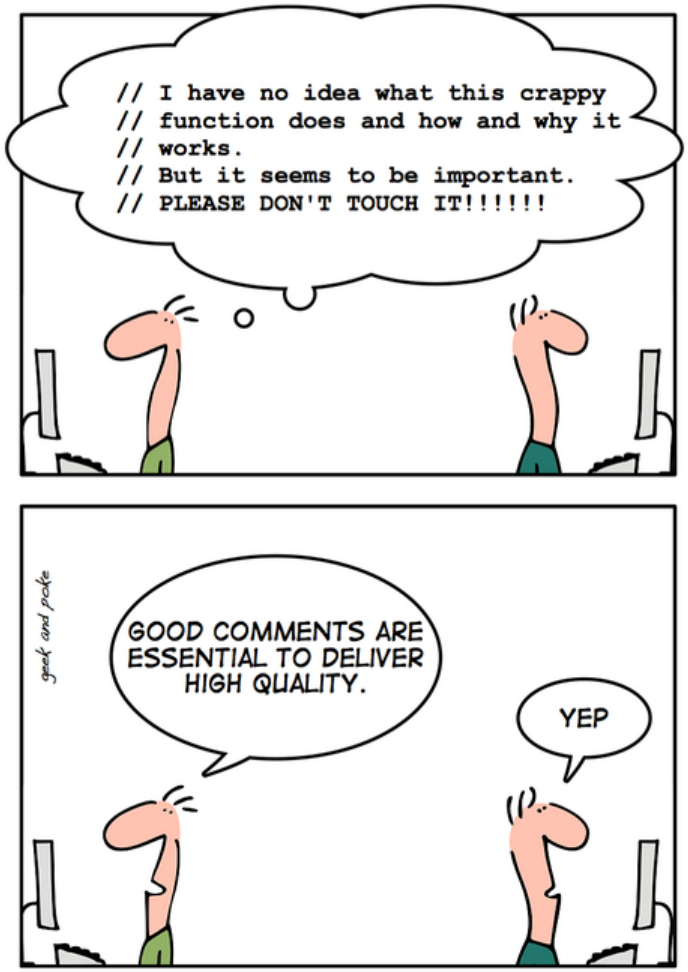
Measure and compare HPC and non-HPC practices

Method

- Create code commit quality measures.
- Derive from the version control systems (VCS)
- Conduct a case study
  - Five key HPC infrastructure frameworks
  - Three highly diverse non-HPC open source projects

Context - Why/Who/What Changed

VCS tracks code (why/who/what changed), allows shared development, and supports complex workflows



Sample of Projects

HPC middlware from ICL at UTK
Diverse non-HPC projects hosted on BitBucket

HPC

1. OpenMPI
2. OpenSHMEM
3. PaRSEC
4. PLASMA
5. MAGMA

Non-HPC

1. Bitbucket Tutorial
2. Django-piston
3. Linux kernel

Approach - Multiple Exploratory Case Study

Literal Replication

Literal replication: five similar widely used parallel computing frameworks

Theoritical Replication

Theoretical replication: three extreme non-HPC projects from Bitbucket

Projects

1. Bitbucket Tutorial
2. Django-piston
3. Linux kernel

1. Most forked
2. Most watched
3. Most commits

Project Summaries: HPC

Different VCS systems:
Git, Mercurial (hg), and SVN for HPC
Git and Mercurial for non-HPC

Repos	**Authors**	**Time**	**Cmts/UCmts**	**VCS**
OpenMPI	116	2003-	20K / 20K	GH-hg
OpenSHMEM	20	2010-	1K / 1K	GH-hg
PaRSEC	33	2009-	8K / 7K	BB-hg
PLASMA	20	2008-	4K / 4K	SVN
MAGMA	21	2013-	4+K / 4-K	SVN

Table : *Overview of HPC projects*

Project Summaries: non-HPC

Selection criteria:

1. The most unique commit comments
2. The most forked
3. The most watched

Repos	**Authors**	**Time**	**Cmts/UCmts**
eniliolopez/linux	15k	2005-	446K / 442K
tutorials.bitbucket	2.6k	2012-	6K / 5.5K
django-piston	33	2009-2012	254 / 252

Table : *Overview of Non-HPC projects*

Measures: Basic

Total Number of Commits

- Effort that went into creating and maintaining the project
- A normalizing factor in commit quality measures

Number of Authors in a project

- A social characteristic of a project
- E.g., commercial projects $\rightarrow$ fewer more equal contributors

Measures: Commit Quality

Number of Unique Commit Messages

- Each commit message should be specific
  - No generic commit messages: "fix," "fixed bug," or "initial commit"
- Mature $\rightarrow$ each commit messages is unique

The size of Commit Comments

- A specific format and detail
- Very small commit messages may indicate immaturity
- Mature $\rightarrow$ larger commit messages

Measures: Commit Quality 2

Number of delta

- The number of files modified or added in a commit
- Although convenient, several tasks in a single commit is bad practice
- Commits with more delta $\rightarrow$ less mature projects

Fraction of Unique Commit Comments

- A high ratio $\rightarrow$ commit comments are tailored to each commit.
- A lower ratio indicates that same comments were reused for new commits or were generic.

Results

Fraction of Unique Commit Comments

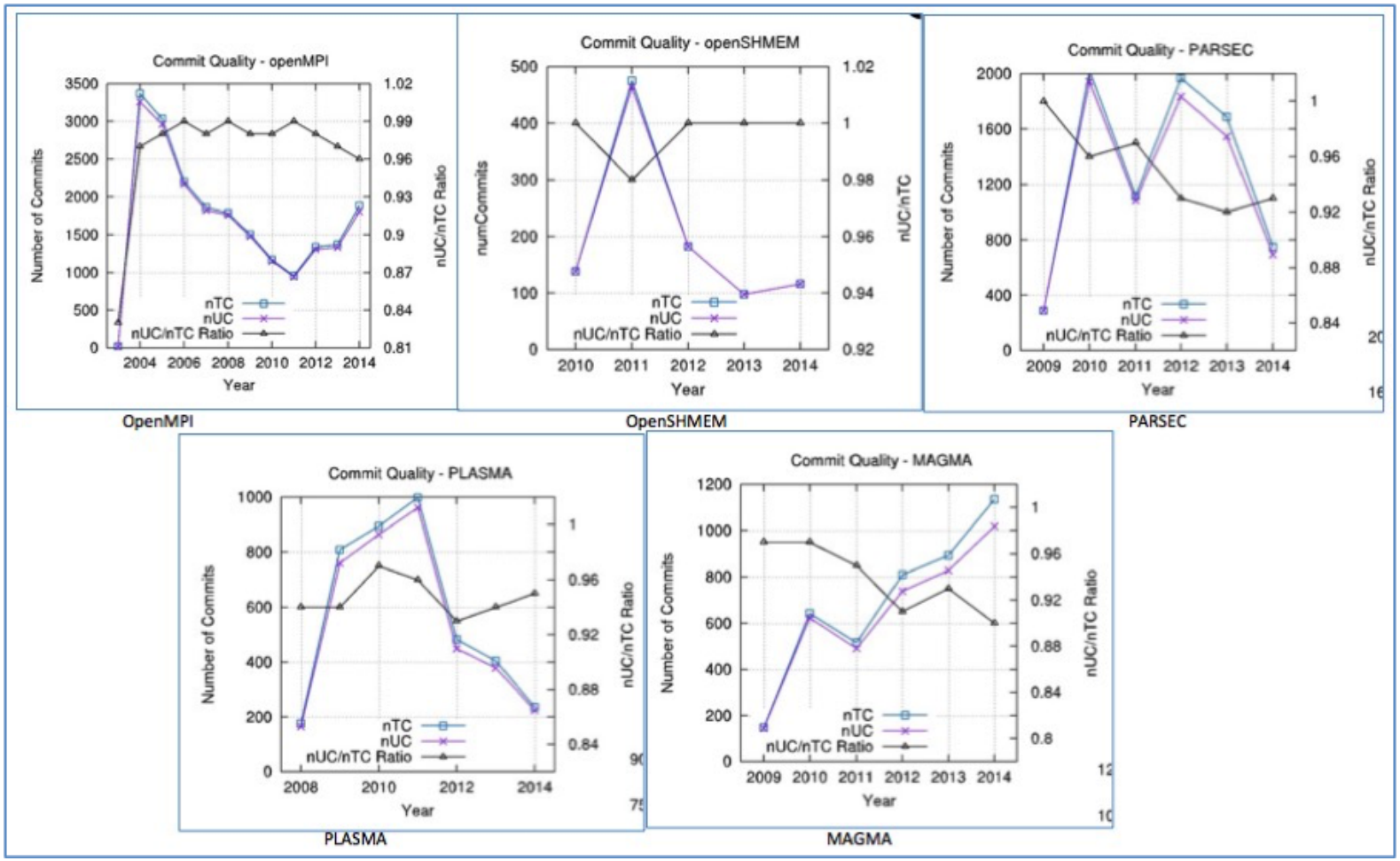


Figure : Trend in commit quality of HPC projects

Results Contd.

Fraction of Unique Commit Comments

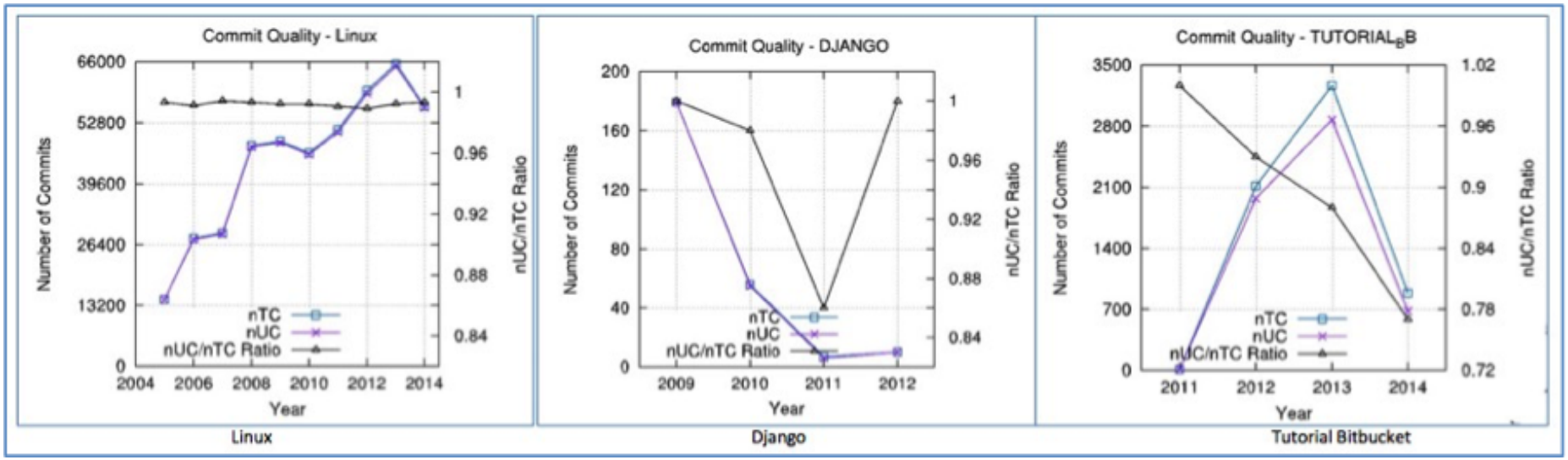


Figure : Trend in commit quality of HPC projects

nUC - Number of Unique Commit Comments

nTC - Total number of Commit Comments

$\frac{nUC}{nTC}$ - Comment Quality

Implications: Frctn of Unique Cmt Cmts

Number of Commits - HPC vs Non-HPC

- Initial spike in number of commits (HPC) $\rightarrow$ the starting activities of the project
- Linux kernel (non-HPC) shows very steady development (no sharp peaks)

Commit Quality - HPC vs Non-HPC

- $\frac{nUC}{nTC} \in [0.9, 1.0] \rightarrow$ effort to document the changes
- $\frac{nUC}{nTC}$ ratio for non-HPC projects less consistent than for HPC
- Average life of five years, Average $\frac{nUC}{nTC} \geq 0.9$

Results Contd.

Number of Delta and Comment Length

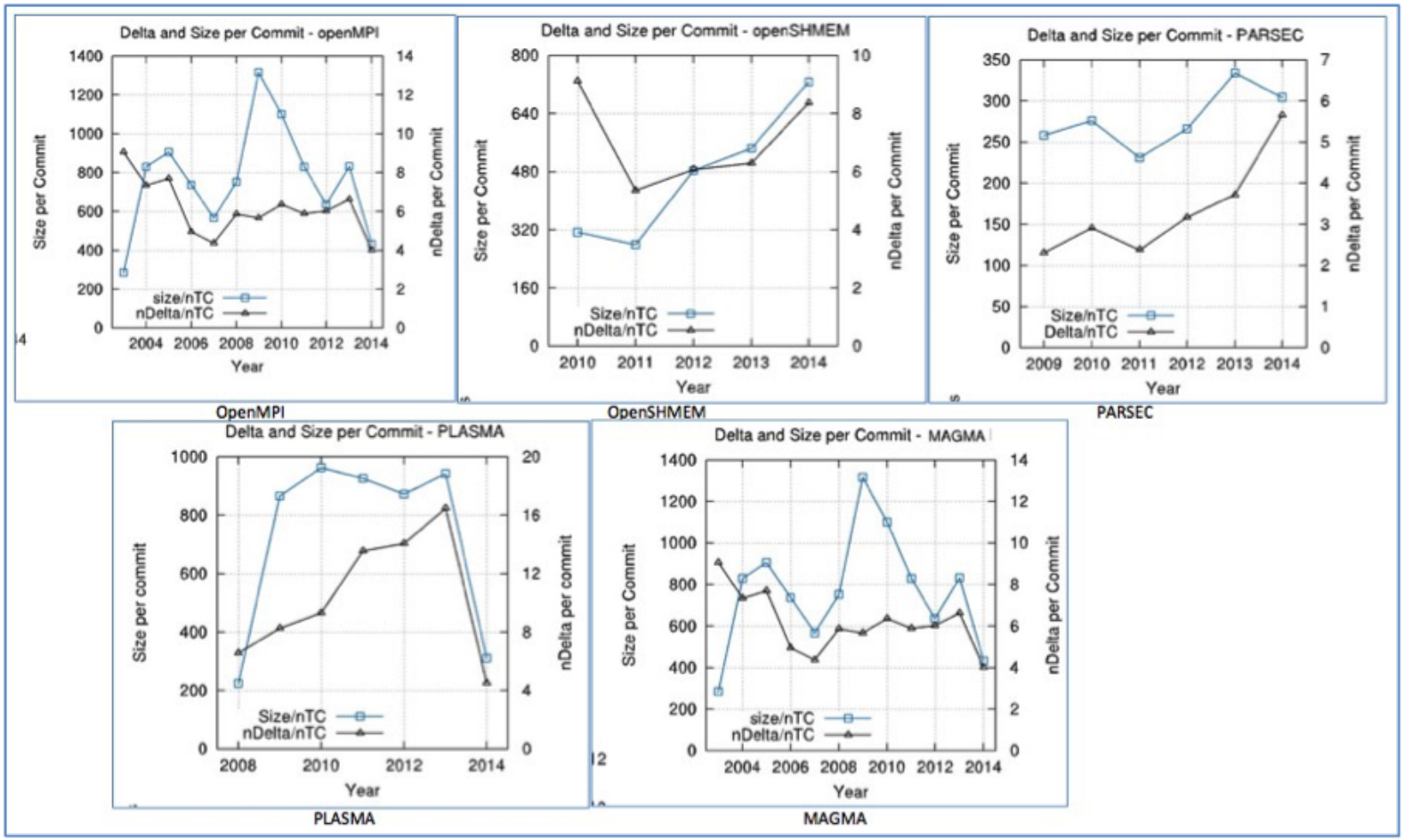


Figure : Size of Commits for HPC projects

Results Contd.

Number of Delta and Comment Length

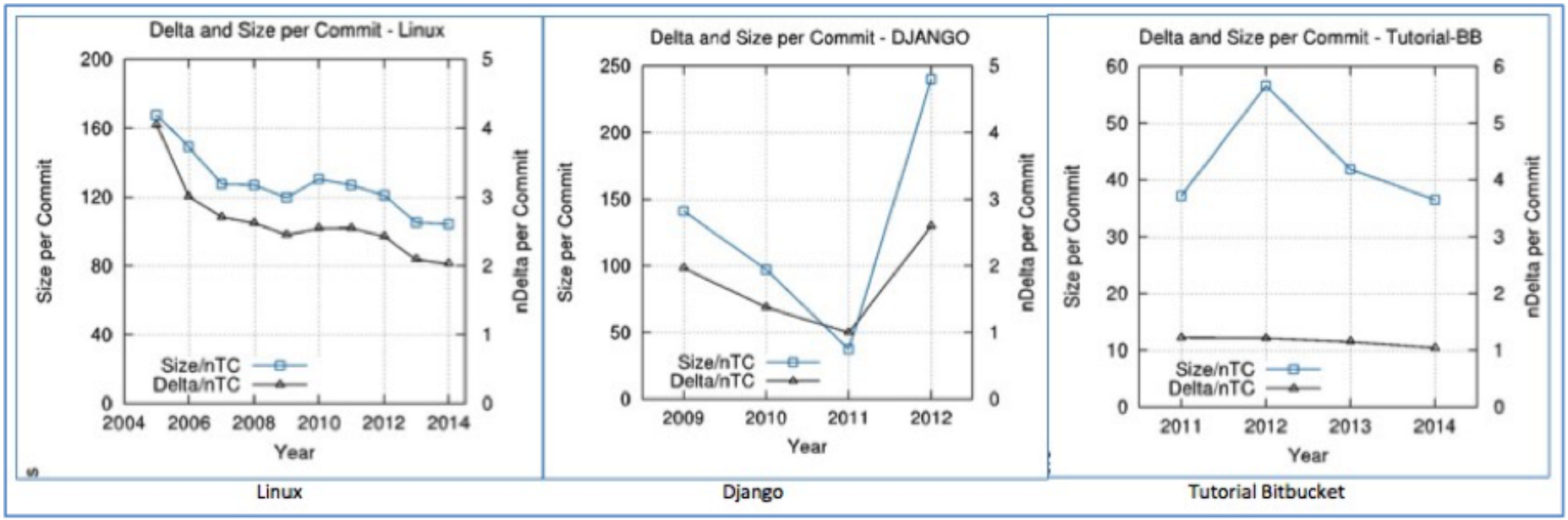


Figure : Size of Commits for Non-HPC projects

Delta - Total Number of files modified or added in a single commit.

Comment Size - Number of character in each commit

Discussion

Commit cComment Size - HPC vs Non-HPC

- HPC projects: 200-1300 characters
- non-HPC projects: 50-150 characters
- More effort in HPC community.

Delta per commit - HPC vs Non-HPC

- HPC: 5-6, up to 9 in PLASMA
- non-HPC: approximately 2.
- More delta per commit:
  - Tangled changes?
  - More complex tasks?

HPC: Higher Quality but More Complex

Observation 1

HPC middleware projects have higher commit quality:

- Fraction of unique commit messages
- Message size

Observation 2

HPC middleware projects have more complex commits:

- The number of files modified in a commit

Conclusion

Despite the HPC community being early in embracing code sharing, it has lagged in efficiently using the tools that define open source development.

The results of our investigation on HPC and other projects suggest the specific hypotheses that we plan to investigate on a more comprehensive set of projects.

Future Work

Are the typical VCS and issue trackers most suitable for HPC development practices? If not, what modifications are needed to make HPC development most productive?

We hope that our initial findings would help pose more precise questions in this area and the methods used would help answer such questions in the future.

Questions?

The End