SC19 BoF
Quality Assurance and Coding Standards for Parallel Software

parallel-code-qa.github.io/sc19-bof

Manuel Arenaz
manuel.arenaz@appentra.com

appentra
make code parallel

Julian Miller
miller@itc.rwth-aachen.de

IT Center | RWTH AACHEN UNIVERSITY

SC19
Denver, CO | hpc is now.

Unión Europea
Fondo Europeo
de Desarrollo Regional
"Una manera de hacer Europa"

Agenda

12:15 - 12:18	*Welcome and introductions (3 minutes)*
12:18 - 12:25	Motivation (7 minutes): How can we lower the cost of adoption of parallel computing?
12:25 - 12:50	Speakers (25 minutes): Discuss the challenges and ideas to address this problem
12:50 - 13:10	Discussion (20 minutes): Interaction with the audience
13:10 - 13:15	*Close (5 minutes)*

Agenda

12:15 - 12:18	***Welcome and introductions (3 minutes)***
12:18 - 12:25	Motivation (7 minutes): How can we lower the cost of adoption of parallel computing?
12:25 - 12:50	Speakers (25 minutes): Discuss the challenges and ideas to address this problem
12:50 - 13:10	Discussion (20 minutes): Interaction with the audience
13:10 - 13:15	*Close (5 minutes)*


David Bernholdt


Saber Feki


Dirk Pleiter


Chunhua Liao


Robert Schiela


Khaled Elamrawi


Manuel Arenaz


Julian Miller

Agenda

12:15 - 12:18	***Welcome and introductions (3 minutes)***
12:18 - 12:25	Motivation (7 minutes): How can we lower the cost of adoption of parallel computing?
12:25 - 12:50	Speakers (25 minutes): Discuss the challenges and ideas to address this problem
12:50 - 13:10	Discussion (20 minutes): Interaction with the audience
13:10 - 13:15	*Close (5 minutes)*

Agenda

12:15 - 12:18	*Welcome and introductions (3 minutes)*
12:18 - 12:25	**Motivation (7 minutes): How can we lower the cost of adoption of parallel computing?**
12:25 - 12:50	Speakers (25 minutes): Discuss the challenges and ideas to address this problem
12:50 - 13:10	Discussion (20 minutes): Interaction with the audience
13:10 - 13:15	*Close (5 minutes)*

Motivation

- The automation of testing is critical in software development to improve quality assurance (QA)
  - but today 80% of testing is manual (Gartner) and $32 billion is spent annually on QA (IDC/Nelson Hall).
- Coding standards in automotive and cybersecurity (e.g. CWE, MISRA) provide developers with rules and recommendations to prevent faulty code patterns.
- The ever-increasing complexity of HPC software and hardware pushes the developers to critically re-evaluate testing methods, but there is no coding standard for parallel programming yet.
- Our goal is to form a community interested in quality assurance and best practices for parallel programming.
- Outcome: List of 5-10 people interested in working on this topic.

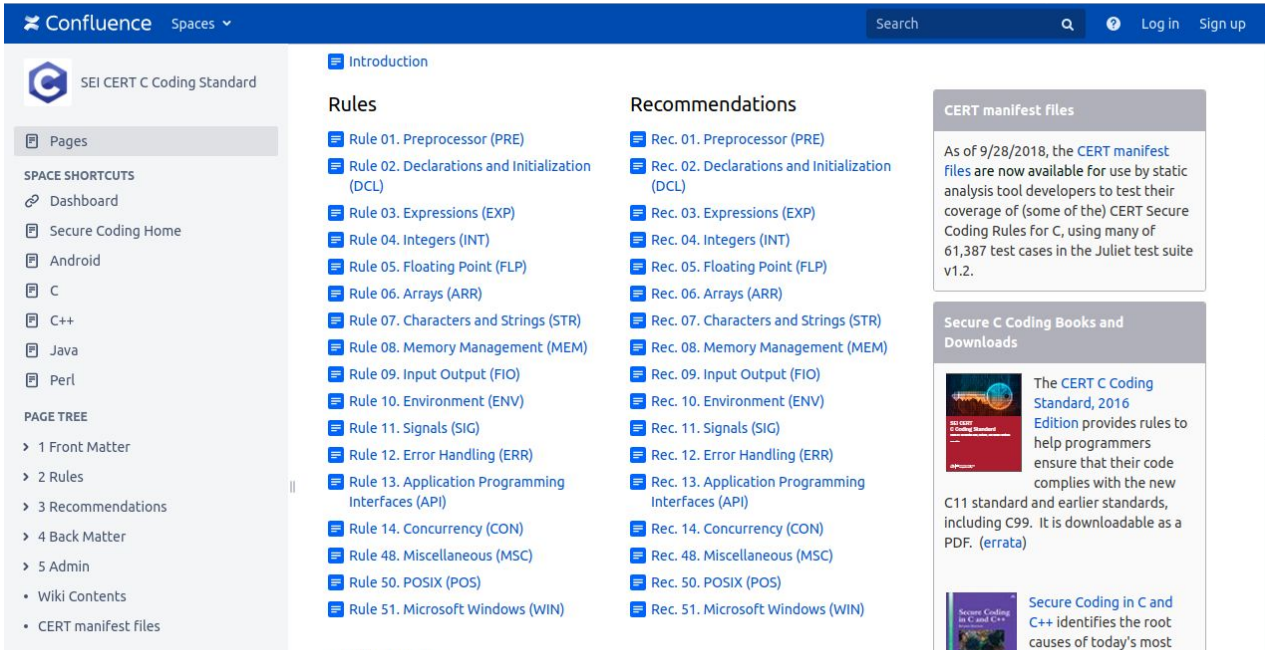
Motivation

> *"Writing programs that scale with increasing numbers of cores should be as easy as writing programs for sequential computers."*
>
> *Asanovic et al. 2009.*

- Develop fast & correct parallel code is a very difficult discipline.
- Because the developer needs a "holistic view" of the code from three perspectives: computations, memory and control flow.

How can we lower the cost of adoption of parallel computing? In other words, how can we make parallel programming easier?

Motivation



- Software Engineering Institute (SEI), Carnegie Mellon University.
- Funded by US Department of Defense (DoD).
  - Reducing Security Weaknesses
  - C, C++, Java
- Coding standards for quality assurance of software.
  - Rules
  - Recommendations

What can we learn from other industries?

Motivation

- Lack of coding standards for parallel programming
- Lack of tools to automate checking compliance of codes with coding standards for parallel programming
- Evidence of real needs for this…
  - Initiative Better Scientific Software (https://bssw.io/)
  - Initiative DataRaceBench at LLNL (https://github.com/LLNL/dataracebench)
  - Initiative at TACC (https://github.com/ritua2/IPT/tree/master/bug-patterns/Bug_Collection)
- We need to learn from other industries (e.g. Automotive, Healthcare, Control Systems, Aerospace)

What is the current status in HPC industry?
How can we move forward as a community?

Agenda

12:15 - 12:18	*Welcome and introductions (3 minutes)*
12:18 - 12:25	Motivation (7 minutes): How can we lower the cost of adoption of parallel computing?
12:25 - 12:50	**Speakers (25 minutes): Discuss the challenges and ideas to address this problem**
12:50 - 13:10	Discussion (20 minutes): Interaction with the audience
13:10 - 13:15	*Close (5 minutes)*


David Bernholdt


Saber Feki


Dirk Pleiter


Chunhua Liao


Robert Schiela


Khaled Elamrawi


Manuel Arenaz


Julian Miller

Panelists

- **The HPC perspective:**
  The importance of developing Better Scientific Software to manage the ever growing complexity of HPC software, and the role of DevOps/CD, reusable components, etc… in increasing the quality of HPC software.
  - David Bernholdt (ORNL, US)
  - Saber Feki (KAUST, Saudi Arabia)
  - Dirk Pleiter(Juelich Supercomputing Center, Germany)

- **Benchmarking Quality Assurance Tools in HPC:**
  Experiences in creating benchmark suites to improve quality assurance and ensure best practices in developing parallel software. DataRaceBench is a benchmark suite designed to systematically and quantitatively evaluate the effectiveness of data race detection tools. It includes a set of OpenMP microbenchmarks with and without data races.
  - Chunhua “Leo” Liao (LLNL, US)

- **Coding standards as a way to facilitate Quality Assurance:**
  Successful experiences providing software developers with rules and recommendations to fix faulty code patterns in functional safety and cybersecurity through coding standards like CWE.
  - Robert Schiela (Software Engineering Institute, Carnegie Mellon University, US)

- **The industry perspective:**
  The importance of Quality Assurance and Testing in HPC, from supercomputing centers to universities, and from startups to Fortune 500 companies.
  - Khaled El-Amrawi (Brightskies)
  - Manuel Arenaz (Appentra Solutions, Spain)

Promoting and Supporting Better Scientific Software

David E. Bernholdt
Oak Ridge National Laboratory
bernholdtde@ornl.gov

This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

The Challenge of Better Scientific Software is Broad

- Our culture values scientific results over the software that produces it
- Consequently, many PIs and research software teams treat their software as a means to an end rather than a tool that they need to continually hone
- The increasing sophistication and scale of the scientific models make the software more challenging to create, sustain, and enhance
- The increasing complexity and diversity of HPC hardware makes the software more challenging to create, sustain, and enhance
- Parallel programming is hard, and unforgiving

Mobilizing to Change the Culture

- A growing number of projects and organizations around the world are recognizing these challenges
  - D. S. Katz *et al*., "Community Organizations: Changing the Culture in Which Research Software Is Developed and Sustained," in *Computing in Science & Engineering*, vol. 21, no. 2, pp. 8-24, 1 March-April 2019. doi: 10.1109/MCSE.2018.2883051
- Using different approaches, targeting different communities
- **Look for opportunities to participate and collaborate!**

Interoperable Design of Extreme-scale Application Software (IDEAS)

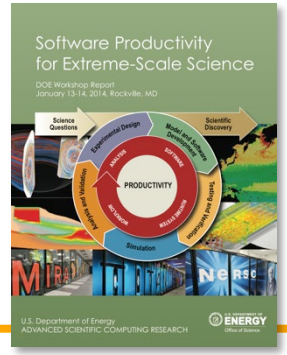
Motivation

Enable ***increased scientific productivity,*** realizing the potential of extreme- scale computing, through ***a new interdisciplinary and agile approach to the scientific software ecosystem***.

Objectives

Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.

Respond to trend of continuous refactoring with efficient agile software engineering methodologies & improved software design.

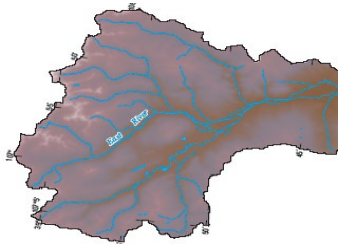


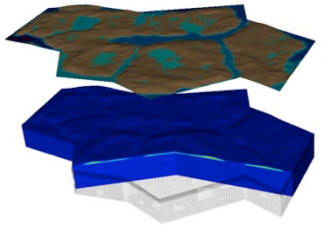
Project History

IDEAS began in 2014 as a DOE ASRC/BER partnership to improve application software productivity, quality, and sustainability. In 2017, the DOE Exascale Computing Project began supporting IDEAS to help application teams improve developer productivity and software sustainability while making major changes for exascale.

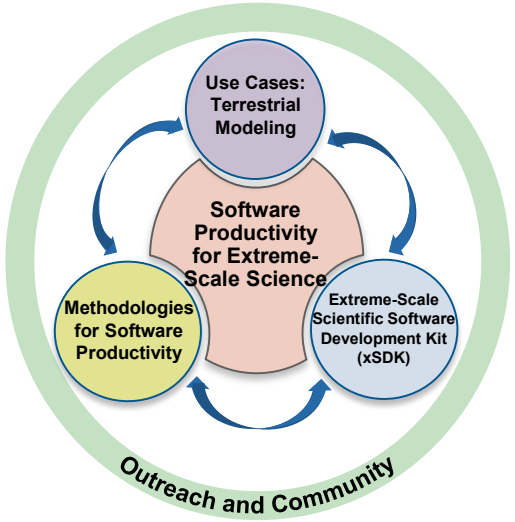
Impact on Applications & Programs

Terrestrial ecosystem use cases tied initial IDEAS activities to programs in DOE Biological and Environmental Research (BER). The Exascale Computing Project (ECP) supports a broad portfolio of applications furthering science, energy, national security, and economic competitiveness.





Approach



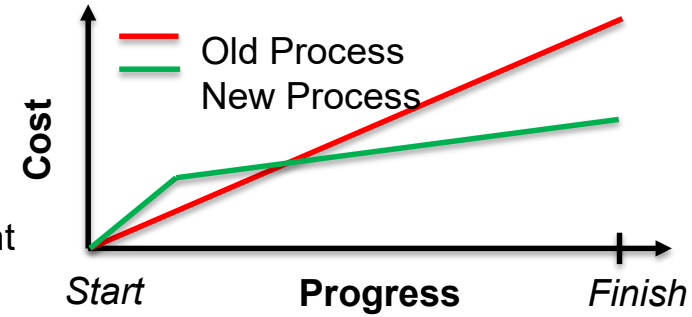
Interdisciplinary multi-institutional team (ANL, LANL, LBNL, LLNL, ORNL, PNNL, SNL, U. Oregon) with broad experience in scientific software development

Close partnerships with applications teams ensures impact on science

Identification, documentation and dissemination of **best practices** for BER and ECP software teams and the broader community

Catalyzing **software process improvements** through tailored engagement with individual projects

Working to bend the curve of software development costs downwards



U.S. DEPARTMENT OF ENERGY | Office of Science

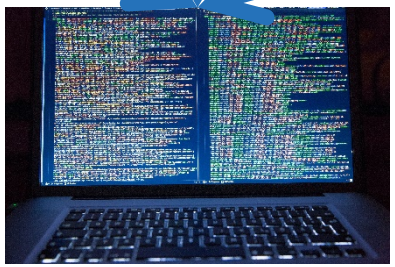
ideas-productivity.org

EXASCALE COMPUTING PROJECT

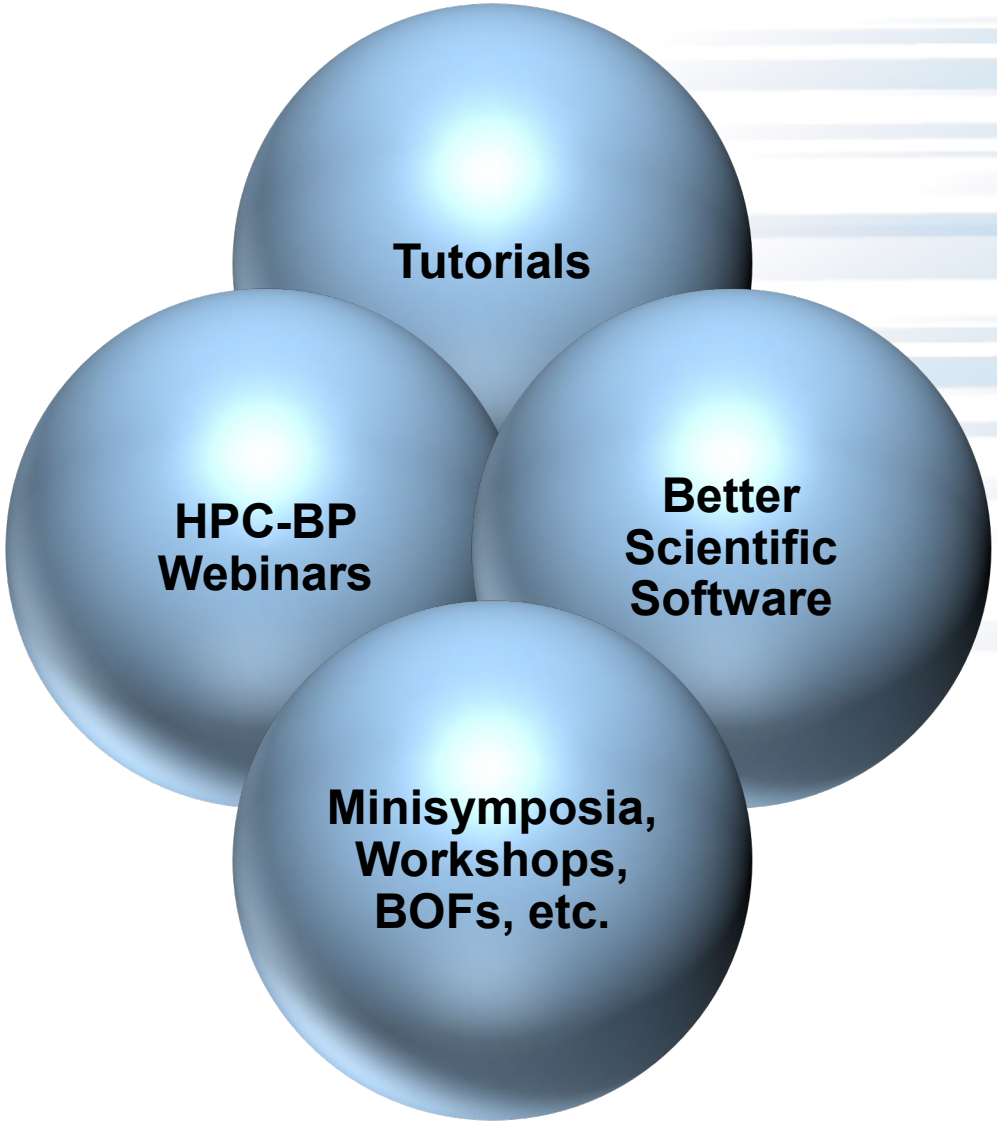
How does IDEAS Achieve Its Goals?

HPC/CSE Community

IDEAS productivity



ECP Code Teams



IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Building an Online Community

https://bssw.io

- **New community-based resource for scientific software improvement**
- A central hub for sharing information on practices, techniques, experiences, and tools to improve developer productivity and software sustainability for computational science & engineering (CSE)

Goals

- Raise awareness of the importance of **good software practices** to scientific productivity and to the quality and reliability of computationally-based scientific results
- Raise awareness of the **increasing challenges** facing CSE software developers as high-end computing heads to extreme scales
- Help CSE researchers **increase effectiveness** as well as leverage and impact
- **Facilitate CSE collaboration via software** in order to advance scientific discoveries

Site users can…

- **Find information** on scientific software topics
- **Contribute new resources** based on your experiences
- Create content tailored to the unique needs and perspectives of a focused scientific domain

Additional Software-Related Events at SC19

Bold events (co-)organized by IDEAS

Day/Time	Event Type	Event Title
Sunday	Tutorial	Floating-Point Analysis and Reproducibility Tools for Scientific Software
Sunday	**Workshop**	**The 2019 International Workshop on Software Engineering for HPC-Enabled Research (SE-HER 2019)**
Monday	**Tutorial**	**Better Scientific Software**
Monday	**Tutorial**	**Managing HPC Software Complexity with Spack**
Monday	Workshop	3nd International Workshop on Software Correctness for HPC Applications (Correctness 2019)
Monday	**Students@SC**	**Students@SC: Modern Software Design, Tools, and Practices**
Tuesday	BoF	Extreme-Scale Scientific Software Stack (E4S)
Tuesday	BoF	Exchanging Best Practices in Supporting Computational and Data-Intensive Research
Tuesday	Panel	Developing and Managing Research Software in Universities and National Labs
Wednesday	**BoF**	**Software Engineering and Reuse in Modeling, Simulation, and Data Analytics for Science and Engineering**
Thursday	BoF	Quality Assurance and Coding Standards for Parallel Software
Thursday	Panel	Sustainability of HPC Research Computing: Fostering Career Paths for Facilitators, Research Software Engineers, and Gateway Creators
Friday	Panel	The Road to Exascale and Beyond is Paved by Software: How Algorithms, Libraries and Tools Will Make Exascale Performance Real

Panelists

- **The HPC perspective:**
  The importance of developing Better Scientific Software to manage the ever growing complexity of HPC software, and the role of DevOps/CD, reusable components, etc… in increasing the quality of HPC software.
  - David Bernholdt (ORNL, US)
  - Saber Feki (KAUST, Saudi Arabia)
  - Dirk Pleiter(Juelich Supercomputing Center, Germany)

- **Benchmarking Quality Assurance Tools in HPC:**
  Experiences in creating benchmark suites to improve quality assurance and ensure best practices in developing parallel software. DataRaceBench is a benchmark suite designed to systematically and quantitatively evaluate the effectiveness of data race detection tools. It includes a set of OpenMP microbenchmarks with and without data races.
  - Chunhua “Leo” Liao (LLNL, US)

- **Coding standards as a way to facilitate Quality Assurance:**
  Successful experiences providing software developers with rules and recommendations to fix faulty code patterns in functional safety and cybersecurity through coding standards like CWE.
  - Robert Schiela (Software Engineering Institute, Carnegie Mellon University, US)

- **The industry perspective:**
  The importance of Quality Assurance and Testing in HPC, from supercomputing centers to universities, and from startups to Fortune 500 companies.
  - Khaled El-Amrawi (Brightskies)
  - Manuel Arenaz (Appentra Solutions, Spain)

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Quality Assurance and Coding Standards for Parallel Software

- System software maintenance and upgrade result in changes in the ecosystem of HPC system
  - Systematic non-regression testing after each maintenance
  - Validate binaries of installed software (3rd party packages), applications and libraries
  - Validate numerical correctness and performance

- Challenge: in-house software validation
  - Most of developers are non-CS
  - Not following standard software engineering best practices
  - Challenging to integrate with non-regression test suite

Panelists

- **The HPC perspective:**
  The importance of developing Better Scientific Software to manage the ever growing complexity of HPC software, and the role of DevOps/CD, reusable components, etc… in increasing the quality of HPC software.
  - David Bernholdt (ORNL, US)
  - Saber Feki (KAUST, Saudi Arabia)
  - Dirk Pleiter(Juelich Supercomputing Center, Germany)

- **Benchmarking Quality Assurance Tools in HPC:**
  Experiences in creating benchmark suites to improve quality assurance and ensure best practices in developing parallel software. DataRaceBench is a benchmark suite designed to systematically and quantitatively evaluate the effectiveness of data race detection tools. It includes a set of OpenMP microbenchmarks with and without data races.
  - Chunhua “Leo” Liao (LLNL, US)

- **Coding standards as a way to facilitate Quality Assurance:**
  Successful experiences providing software developers with rules and recommendations to fix faulty code patterns in functional safety and cybersecurity through coding standards like CWE.
  - Robert Schiela (Software Engineering Institute, Carnegie Mellon University, US)

- **The industry perspective:**
  The importance of Quality Assurance and Testing in HPC, from supercomputing centers to universities, and from startups to Fortune 500 companies.
  - Khaled El-Amrawi (Brightskies)
  - Manuel Arenaz (Appentra Solutions, Spain)

SC19 BOF: QUALITY ASSURANCE AND CODING STANDARDS FOR PARALLEL SOFTWARE

Dirk Pleiter | SC'19, Denver | 21.11.2019

JÜLICH
Forschungszentrum

Why is Quality Assurance Increasingly Important?

Increasing complexity of application increases risk of introducing errors

- Quality assurance procedures help avoiding introducing (and keeping undetected) errors

Code correctness and standard compliance becomes more important when code has to run on a variety of architectures

- Errors may show-up when moving to other architectures

Accomplish code refactoring for exascale systems efficiently

- Need to keep "technical debt" low

Write code that allows to leverage different levels of parallelism

- Example: Robust frameworks facilitating data layout transformations

Digression: Technical Debt

Technical debt as a metaphor: Financial debt vs. future costs associated to fixing immature software

- Economic concept

Aim for developers being aware of the technical debt that they incur during application development

- Aim for continues assessment of technical debt, e.g. application of coding quality rules

Technical debt management tools

- Based on static code analysis
- Technical debt assessment
  - Systematic approach to prioritise code updates
  - Approach to jointly defining and prioritising code standards

How to Promote Quality Assurance?

Develop/enhance/promote uptake of relevant tools

- Need to establish benefits for code developers

Involve software engineers in development of scientific applications

- Very hard to achieve due to lacking attractive career paths

Improve education and training

- "Software Development in Science" is often not part of curricula at universities or HPC training programs

Panelists

- **The HPC perspective:**
  The importance of developing Better Scientific Software to manage the ever growing complexity of HPC software, and the role of DevOps/CD, reusable components, etc… in increasing the quality of HPC software.
  - David Bernholdt (ORNL, US)
  - Saber Feki (KAUST, Saudi Arabia)
  - Dirk Pleiter(Juelich Supercomputing Center, Germany)

- **Benchmarking Quality Assurance Tools in HPC:**
  Experiences in creating benchmark suites to improve quality assurance and ensure best practices in developing parallel software. DataRaceBench is a benchmark suite designed to systematically and quantitatively evaluate the effectiveness of data race detection tools. It includes a set of OpenMP microbenchmarks with and without data races.
  - Chunhua “Leo” Liao (LLNL, US)

- **Coding standards as a way to facilitate Quality Assurance:**
  Successful experiences providing software developers with rules and recommendations to fix faulty code patterns in functional safety and cybersecurity through coding standards like CWE.
  - Robert Schiela (Software Engineering Institute, Carnegie Mellon University, US)

- **The industry perspective:**
  The importance of Quality Assurance and Testing in HPC, from supercomputing centers to universities, and from startups to Fortune 500 companies.
  - Khaled El-Amrawi (Brightskies)
  - Manuel Arenaz (Appentra Solutions, Spain)

Benchmarking Quality Assurance Tools

- Evaluating Data Race Detection tools using DataRaceBench

Chunhua “Leo” Liao, Pei-Hung Lin , Markus Schordan and Ian Karlin

SC19: BoF Quality Assurance and Coding Standards for Parallel Software
Nov. 21, 2019, Denver, CO

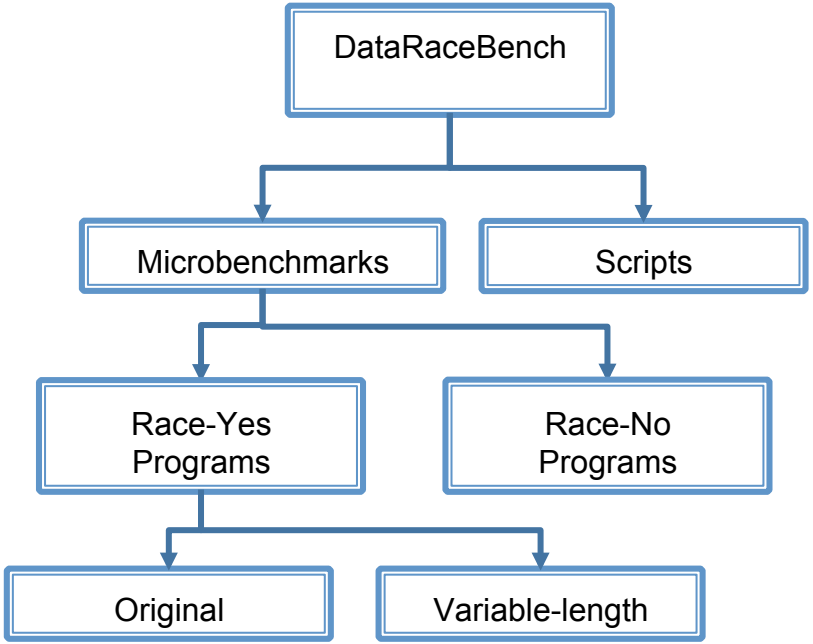
LLNL-PRES-796717

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

DataRaceBench: a Dedicated Data Race Detection Benchmark Suite

- Motivation
  - the lack of apple-to-apple comparison among data race detection tools
- Containing both positive and negative tests
  - Generate standard quantitative metrics
  - Precision, recall, and accuracy
- Coverage: 116 total microbenchmarks
  - V1.0.1: 72 from AutoPar's regression tests, PolyOpt, LLNL apps, etc.
  - v1.2.0: 44 more based on semantics coverage analysis of OpenMP 4.5



https://github.com/LLNL/dataracebench

Design Philosophy: Positive + Negative Tests

```
1.      …
2.      int i,x;
3.      #pragma omp parallel for
4.       for (i=0;i<100;i++)
5.       {   x=i;  }
6.       printf("x=%d",x);
7.      …
```

lastprivatemissing-orig-yes.c

one data race pair
x@5 vs. x@5
Y2: Missing data sharing clauses

```
1.    …
2.    int i,x;
3.    #pragma omp parallel for lastprivate (x)
4.    for (i=0;i<100;i++)
5.     {   x=i; }
6.    printf("x=%d",x);
7.    …
```

lastprivate-orig-no.c

N2: Use of data sharing clauses

V1.2.0 Report for Archer and Intel Inspector

Microbenchmark Program	R	Data Race Detection Tools					
		Archer			Intel Inspector		
		min race	max race	type	min race	max race	type
DRB073-doall2-orig-yes.c	Y	84	92	TP	2	2	TP
DRB074-flush-orig-yes.c	Y	1	3	TP	1	1	TP
DRB075-getthreadnum-orig-yes.c	Y	71	71	TP	1	1	TP
DRB076-flush-orig-no.c	N	0	0	TN	0	0	TN
DRB077-single-orig-no.c	N	0	0	TN	0	0	TN
DRB078-taskdep2-orig-no.c	N	0	0	TN	0	0	TN
DRB079-taskdep3-orig-no.c	N	0	0	TN	0	0	TN
DRB080-func-arg-orig-yes.c	Y	71	71	TP	1	1	TP
DRB081-func-arg-orig-no.c	N	0	0	TN	0	0	TN
DRB082-declared-in-func-orig-yes.c	Y	71	71	TP	1	1	TP
DRB083-declared-in-func-orig-no.c	N	0	0	TN	0	0	TN
DRB084-threadprivatemissing-orig-yes.c	Y	71	71	TP	1	1	TP
DRB085-threadprivate-orig-no.c	N	-		CSF	0	0	TN
DRB086-static-data-member-orig-yes.cpp	Y	-		CSF	1	1	TP
DRB087-static-data-member2-orig-yes.cpp	Y	-		CSF	1	1	TP
DRB088-dynamic-storage-orig-yes.c	Y	71	71	TP	1	1	TP
DRB089-dynamic-storage2-orig-yes.c	Y	71	71	TP	1	1	TP
DRB090-static-local-orig-yes.c	Y	71	71	TP	1	1	TP
DRB091-threadprivate2-orig-no.c	N	-		CSF	0	0	TN
DRB092-threadprivatemissing2-orig-yes.c	Y	71	71	TP	1	1	TP
DRB093-doall2-collapse-orig-no.c	N	0	0	TN	0	0	TN
DRB094-doall2-ordered-orig-no.c	N	-		CUN	0	0	RTO

Microbenchmark Program	R	Data Race Detection Tools					
		Archer			Intel Inspector		
		min race	max race	type	min race	max race	type
DRB095-doall2-taskloop-orig-yes.c	Y	-		CUN	2	2	TP
DRB096-doall2-taskloop-collapse-orig-no.c	N	-		CUN	0	4	FP TN
DRB097-target-teams-distribute-orig-no.c	N	0	0	RSF	0	0	TN
DRB098-simd2-orig-no.c	N	0	0	TN	0	0	TN
DRB099-targetparallelfor2-orig-no.c	N	0	0	TN	0	0	TN
DRB100-task-reference-orig-no.cpp	N	-		CUN	0	0	TN
DRB101-task-value-orig-no.cpp	N	0	0	TN	0	0	TN
DRB102-copyprivate-orig-no.c	N	-		CSF	0	0	TN
DRB103-master-orig-no.c	N	0	0	TN	0	0	TN
DRB104-nowait-barrier-orig-no.c	N	0	0	TN	0	0	TN
DRB105-taskwait-orig-no.c	N	0	0	TN	3	4	FP
DRB106-taskwaitmissing-orig-yes.c	Y	35	48	RTO	TP	4	6	TP
DRB107-taskgroup-orig-no.c	N	0	0	TN	1	1	FP
DRB108-atomic-orig-no.c	N	0	0	TN	0	0	TN
DRB109-orderedmissing-orig-yes.c	Y	71	71	TP	1	1	TP
DRB110-ordered-orig-no.c	N	0	0	TN	0	0	TN
DRB111-linearmissing-orig-yes.c	Y	73	85	TP	1	2	TP
DRB112-linear-orig-no.c	N	-		CUN	0	0	TN
DRB113-default-orig-no.c	N	0	0	TN	0	0	TN
DRB114-if-orig-yes.c	Y	42	48	TP	1	1	TP
DRB115-forsimd-orig-yes.c	Y	44	47	TP	1	1	TP
DRB116-target-teams-orig-yes.c	Y	0	0	RSF	1	1	TP

Compile-time seg. fault (CSF),
Unsupported feature (CUN)

Runtime seg. Fault (RSF),
Runtime timeout (RTO)

Latest regression results (partial)

ID	R	Tool-Compiler														
		Arch.1-Cl.391	Arch.2-Cl.600	Ins.18-In.1702	Ins.18-In.1802	Ins.18-In.1900	Ins.18-In.1904	Ins.19-In.1702	Ins.19-In.1802	Ins.19-In.1900	Ins.19-In.1904	ROMP-Cl.800	Tsan5-Cl.502	Tsan6-Cl.601	Tsan7-Cl.710	Tsan8-Cl.801
64	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
65	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	RTO	✓	✓	✓	✓
66	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
67	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
68	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
71	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
72	N	✓	✓	✓/✗	✓/✗	✓	✓/✗	✓	✓	✓	✓/✗	✓	✓	✓	✓/✗	✓/✗
75	Y	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓	✓	✓
80	Y	✓/✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
82	Y	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗	✓/✗	✓/✗
85	N	CSF	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓
86	Y	CSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗
87	Y	CSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗	✓/✗	✓/✗
91	N	CSF	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓
93	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
94	N	CUN	CUN	CSF	RSF	RSF	✓	CSF	RSF	RSF	✓	CUN	CUN	CUN	CUN	CUN
95	Y	CUN	CUN	✓	✓	✓	✓	✓	✓	✓	✓	CUN	CUN	CUN	CUN	CUN
96	N	CUN	CUN	✓/✗	✓	✗	✓	✓	✓	✗	✓	CUN	CUN	CUN	CUN	CUN
97	N	RSF	✗	✓	✓	✗	✓	✓	✓	✗	✓	CUN	RTO	✓	✓	CUN
99	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
100	N	CUN	CUN	✓	✓	✓	✓	✓	✓	✓	✓	CUN	CUN	CUN	CUN	CUN
102	N	CSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
105	N	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	RSF	✓	✓	✓	✓
106	Y	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	RSF	✓	✓	✓	✓
107	N	✓	✓	✗	✗	✗	✓	✗	✗	✓/✗	✓	✓	✓	✓	✓	✓
108	N	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
110	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗
112	N	CUN	CUN	CSF	✓	✗	✓	CSF	✓	✗	✓	CUN	CUN	CUN	CUN	CUN
113	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
114	Y	✓	✓/✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓	✓	✓/✗
116	Y	RSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	RSF	✓	✓	✓	✓

https://github.com/LLNL/dataracebench/wiki/Regression-metrics

Compile-time seg. fault (CSF)
Unsupported feature (CUN)
Runtime seg. Fault (RSF)
Runtime timeout (RTO)

Take-away Messages

- Quality assurance needs tools, a lots of them.
  - But who are watching the quality of these tools?
- Regression tests (benchmarks) are critical to measure and improve the quality of any software, including quality assurance tools.
  - Must have both Positive and Negative tests
- DataRaceBench is a set of regression test for data race detection tools.
  - Successfully identified limitations and bugs for popular tools, sent actionable info. to developers
  - Showing regression of tools across different versions
  - Providing a dashboard summarizing the state-of-the-art of data race detection
- Maintaining a tool benchmark suite (regression tests) is challenging.
  - Constant and fast evolving of parallel language standards
  - Largely manual process to analyze and improve coverage of language features and code patterns
  - Resources (labor+machines) to regularly run and publish results

https://github.com/LLNL/dataracebench

Panelists

- **The HPC perspective:**
  The importance of developing Better Scientific Software to manage the ever growing complexity of HPC software, and the role of DevOps/CD, reusable components, etc… in increasing the quality of HPC software.
  - David Bernholdt (ORNL, US)
  - Saber Feki (KAUST, Saudi Arabia)
  - Dirk Pleiter(Juelich Supercomputing Center, Germany)

- **Benchmarking Quality Assurance Tools in HPC:**
  Experiences in creating benchmark suites to improve quality assurance and ensure best practices in developing parallel software. DataRaceBench is a benchmark suite designed to systematically and quantitatively evaluate the effectiveness of data race detection tools. It includes a set of OpenMP microbenchmarks with and without data races.
  - Chunhua “Leo” Liao (LLNL, US)

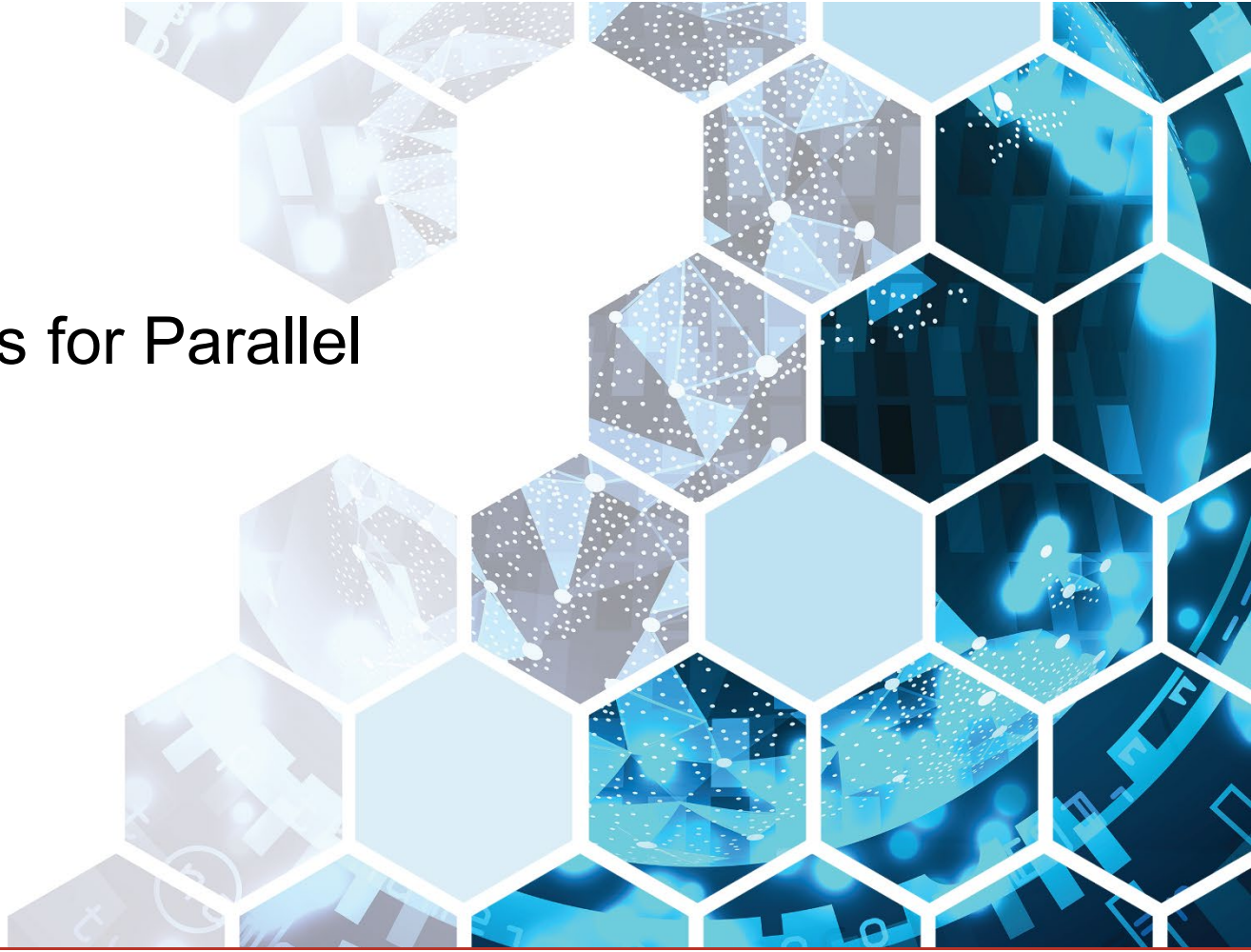
- **Coding standards as a way to facilitate Quality Assurance:**
  Successful experiences providing software developers with rules and recommendations to fix faulty code patterns in functional safety and cybersecurity through coding standards like CWE.
  - Robert Schiela (Software Engineering Institute, Carnegie Mellon University, US)

- **The industry perspective:**
  The importance of Quality Assurance and Testing in HPC, from supercomputing centers to universities, and from startups to Fortune 500 companies.
  - Khaled El-Amrawi (Brightskies)
  - Manuel Arenaz (Appentra Solutions, Spain)

Coding Standards for Parallel Software

Robert Schiela

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM19-1226

Dimensions for Developing Coding Standards

What is the purpose? What Qualities to Assure?

- Performance!!!
- Safety? – MISRA & AUTOSAR
- Security? – CWE, OWASP & CERT Secure Coding Standards
- Complexity, Maintainability, Readability? – Style Standards

How specific will our standards be?

- Design vs Code (Framework/Library Abstractions?)
- Function vs Form (Security Software vs. Software Security)
- Language agnostic vs. language specific constructs

How will we define patterns? – Identify anti-patterns

- Security: Vulnerabilities, Undefined Behavior, Abstraction Layer Interfaces

How will it be used? – Prevention, Enforcement, Tools, Automation

Parallel Software Concerns?

Performance!!!

But

- Software Provenance
- Information Assurance (Confidentiality, Integrity, Availability)
- Privacy
- Functional Safety for Safety Critical

And

- Robust Distribution and Recombination
- Heterogeneous → Portability or Virtualized Abstraction
- AI & Deep Learning
  - Untrusted, Unstructured Data – Hard to Validate Input
  - Untrusted Implementation – Lack of Explainability = Hard to Verify Results)

Abstraction Layer Interfaces (misinterpretation leading to defects)

Node/Host Layers (Compilation Errors like Undefined Behavior, Injection Attacks)

- Interdependent Services/Applications
- High-Level Software (Pre-Compiled/Interpreted)
- Low-Level Software (Post-Compiled/Interpreted)
- Hardware
- Virtualization & Software-defined hardware blur the lines

Server Stack Layers (Injection Attacks)

- Client / Front-end Server / Backend Server)

System of Systems (Injection Attacks, Reliability)

- Dependencies on external services and data

Network Stack Layers and Protocols

Data vs. Instruction (Untrusted Inputs affecting data flow & control flow)

For More Information

Robert Schiela

Technical Manager

Secure Coding

Email: rschiela@cert.org

Web:

securecoding.cert.org (wiki)

www.cert.org/secure-coding

Robert leads the Secure Coding group at the SEI in defining and executing research and transitioning knowledge that improves the state of the art and practice in secure software development.

Robert has worked in the fields of IT, software engineering, and software education for more than 20 years. He has been at the SEI for nearly 15 years, primarily focused on software process, quality, and security.

Panelists

- **The HPC perspective:**
  The importance of developing Better Scientific Software to manage the ever growing complexity of HPC software, and the role of DevOps/CD, reusable components, etc… in increasing the quality of HPC software.
  - David Bernholdt (ORNL, US)
  - Saber Feki (KAUST, Saudi Arabia)
  - Dirk Pleiter(Juelich Supercomputing Center, Germany)

- **Benchmarking Quality Assurance Tools in HPC:**
  Experiences in creating benchmark suites to improve quality assurance and ensure best practices in developing parallel software. DataRaceBench is a benchmark suite designed to systematically and quantitatively evaluate the effectiveness of data race detection tools. It includes a set of OpenMP microbenchmarks with and without data races.
  - Chunhua “Leo” Liao (LLNL, US)

- **Coding standards as a way to facilitate Quality Assurance:**
  Successful experiences providing software developers with rules and recommendations to fix faulty code patterns in functional safety and cybersecurity through coding standards like CWE.
  - Robert Schiela (Software Engineering Institute, Carnegie Mellon University, US)

- **The industry perspective:**
  The importance of Quality Assurance and Testing in HPC, from supercomputing centers to universities, and from startups to Fortune 500 companies.
  - Khaled El-Amrawi (Brightskies)
  - Manuel Arenaz (Appentra Solutions, Spain)

SC19 BoF - Quality Assurance and Coding Standards for Parallel Software

Manuel Arenaz
manuel.arenaz@appentra.com

Unión Europea
Fondo Europeo
de Desarrollo Regional
"Una manera de hacer Europa"

appentra
make code parallel

Metrics for Parallelization of Codes

No.	Value	Metric
1	**Accelerate the runtime of the software** - Legacy software that needs to be modernised to run on modern hardware - sequential software to be parallelized - parallel software to exploit more parallelism - parallel software ported to a different hardware (eg. CPU to GPU)	**Speedup**
2	**Train customer's workforce** - developer/maintainer learns and enforces best practices for development of parallel software - development/maintainer applies code changes to clearly understand their value from the point of view of parallelization	**Defects** (eg. race conditions) **Recommendations** (eg. best practices) **Parallelization opportunities** **Parallelization strategies** **Parallelization tools**

Capabilities of Parallelware tools

- Detection of defects in parallel code, i.e. race conditions not detected yet
  - Static data race detection for multicore CPUs and GPUs
  - See https://www.appentra.com/knowledge/checks/
- Enforce best practices for parallel programming through suggestions for code refactorizations
  - See https://www.appentra.com/knowledge/checks/
- Discover parallelization opportunities through in-depth static code analysis
  - Source code analysis guided through code patterns, not line by line
  - See https://www.appentra.com/knowledge/patterns/
- Quickly design and implement parallel code for CPU/GPU using OpenMP/OpenACC
  - Parallel code implementation guided by code patterns, not “happy idea”
  - See https://www.appentra.com/knowledge/patterns/

Appentra approach is to break down best practices into items that are objective, measurable and actionable

Knowledge Base of Defects & Recommendations

https://www.appentra.com/knowledge/checks/

Static code analysis tools are designed to aid software developers to build better quality software in less time, by promoting best practices and detecting defects early in the software development life cycle. A defect can lead to a minor malfunction or cause serious security and safety issues. Thus, identifying, mitigating and resolving defects is an essential part of the software development process.

Parallelware Analyzer reports code defects that constitute errors and issues recommendations to adopt best practices that prevent errors related to concurrency and parallelism.

Defects

- **PWD001**: Invalid OpenMP multithreading datascoping
- **PWD002**: Unprotected multithreading reduction operation

Recommendations

- **PWR001**: Declare global variables as function parameters
- **PWR002:** Declare scalar variables in the smallest possible scope
- **PWR003**: Explicitly declare pure functions
- **PWR004**: Declare OpenMP scoping for all variables
- **PWR005**: Disable default OpenMP scoping
- **PWR006**: Avoid privatization of read-only variables

PWD001: Invalid OpenMP multithreading datascoping

Definition

A variable is not being correctly handled in the OpenMP multithreading datascoping clauses.

Relevance

Specifying an invalid scope for a variable will most likely introduce a race condition, making the result of the code unpredictable. For instance, when a variable is written from parallel threads and the specified scoping is shared instead of private.

Actions

Set the proper scope for the variable.

Code example

The following code inadvertently shares the inner loop index variable *j* for all threads, which creates a race condition. This happens because a scoping has not been specified and it will be shared by default.

```
1  void foo() {
2   int result[10][10];
3   int i, j;
4
5   #pragma omp parallel for shared(result)
6   for (i = 0; i < 10; i++) {
7     for (j = 0; j < 10; j++) {
8       result[i][j] = 0;
9     }
10  }
11 }
```

To fix this, the j variable must be declared private. The following code also specifies a private scope for i although in this case it is redundant since OpenMP automatically handles the parallelized loop index variable.

```
1  void foo() {
2   int result[10][10];
3   int i, j;
4
5   #pragma omp parallel for shared(result) private(i, j)
6   for (i = 0; i < 10; i++) {
7     for (j = 0; j < 10; j++) {
8       result[i][j] = 0;
9     }
10  }
11 }
```

PWR002: Declare scalar variables in the smallest possible scope

Code example

In the following code, the function *foo* declares a variable *t* used in each iteration of the loop to hold a value that is then assigned to the array *result*. The variable *t* is not used outside of the loop.

```
1  void foo() {
2    int t;
3    int result[10];
4
5    for (int i = 0; i < 10; i++) {
6      t = i + 1;
7      result[i] = t;
8    }
9  }
```

In this code, the smallest possible scope for the variable *t* is within the loop body. The resulting code would be as follows:

```
1  void foo() {
2    int result[10];
3
4    for (int i = 0; i < 10; i++) {
5      int t = i;
6      result[i] = t + 1;
7    }
8  }
```

From the perspective of parallel programming, moving the declaration of variable *t* to the smallest possible scope helps to prevent potential race conditions. For example, in the OpenMP parallel implementation shown below there is no need to use the clause *private(t)*, as the declaration scope of *t* inherently dictates that it is private to each thread. This avoids potential race conditions because each thread modifies its own copy of the variable *t*.

```
1  void foo() {
2    int result[10];
3
4    #pragma omp parallel for default(none) shared(result)
5    for (int i = 0; i < 10; i++) {
6      int t = i;
7      result[i] = t + 1;
8    }
9  }
```

Resources related to coding guidelines

- G.J. Holzmann (2006-06-19). "The Power of 10: Rules for Developing Safety-Critical Code". *IEEE Computer*. **39** (6): 95–99. doi:10.1109/MC.2006.212. See Rule 6: "Declare all data objects at the smallest possible level of scope". [last checked May 2019]

NAS Parallel Benchmarks: “pwcheck” summary

```
(base) javi@javi-u18:~$ pwcheck --summary NPB3.3-SER-C/ -- -I NPB3.3-SER-C/common/
Compiler flags: -I NPB3.3-SER-C/common/

Found a total of 1160 checks in 80 files successfully analyzed and 1 failures in 4472 ms:

CODE COVERAGE
  Analyzable files:              80 / 81 (98.77 %)
  Analyzable functions:          62 / 391 (15.86 %)
  Analyzable loops:             634 / 1343 (47.21 %)

SUMMARY
  Total defects:                  0
  Total recommendations:       1160
  Total opportunities:         465

SUGGESTIONS

  709 loops could not be analyzed, run pwloops to get more information about them:
        pwloops --non-analyzable NPB3.3-SER-C/ -- -I NPB3.3-SER-C/common/

  1160 recommendations were found in your code, re-run pwcheck without --summary to get details:
        pwcheck --only-recommendations NPB3.3-SER-C/ -- -I NPB3.3-SER-C/common/

  465 opportunities for parallelization were found in your code, run pwloops to get more information about them:
        pwloops NPB3.3-SER-C/ -- -I NPB3.3-SER-C/common/
```

Started collaboration with LLNL: DataRaceBench

	Total	TP	TN	FP	FN	TU	FU	Errors	Time	Correctness success	Precision	Recall	Accuracy
DRACO	116	26	20	0	0	28	33	9	00:14:17	0.4	0.58	0.55	0.59
Parallelware Analyzer	116	3	40	1	45	8	14	5	00:00:06	0.37	0.25	0.05	0.39
Archer 2.0	386	202	156	3	20	-	-	5	00:06:17	0.90	0.99	0.91	0.94
Inspector 19/1904	396	195	164	7	30	-	-	0	01:37:27	0.91	0.97	0.87	0.91
ThreadSanitizer LLVM8	384	184	152	4	38	-	-	6	00:07:03	0.81	0.98	0.83	0.89
ROMP	384	198	144	6	18	-	-	18	00:59:20	0.85	0.97	0.92	0.93

TP: reported existent race conditions
FP: reported non-existent race conditions
TN: didn't report any race conditions when non exist
FN: failed to report existent race conditions
TU: reported unknown for existent race conditions
FU: reported unknown for non-existent race conditions

Errors: tool crashes or failed analyses
Correctness success: TP + TN / Total
Precision: TP / (TP + FP)
Recall: TP / (TP + FN)
Accuracy: (TP + TN) / (TP + TN + FP + FN)

SC19 BoF - Quality Assurance and Coding Standards for Parallel Software

Manuel Arenaz
manuel.arenaz@appentra.com

Unión Europea
Fondo Europeo
de Desarrollo Regional
"Una manera de hacer Europa"

appentra
make code parallel

Agenda

12:15 - 12:18	*Welcome and introductions (3 minutes)*
12:18 - 12:25	Motivation (7 minutes): How can we lower the cost of adoption of parallel computing?
12:25 - 12:50	Speakers (25 minutes): Discuss the challenges and ideas to address this problem
12:50 - 13:10	**Discussion (20 minutes): Interaction with the audience**
13:10 - 13:15	*Close (5 minutes)*

Discussion

How can we lower the cost of adoption of parallel computing?
In other words, how can we make parallel programming easier?

What can we learn from other industries?

What is the current status in HPC industry?
How can we move forward as a community?

Agenda

12:15 - 12:18	*Welcome and introductions (3 minutes)*
12:18 - 12:25	Motivation (7 minutes): How can we lower the cost of adoption of parallel computing?
12:25 - 12:50	Speakers (25 minutes): Discuss the challenges and ideas to address this problem
12:50 - 13:10	Discussion (20 minutes): Interaction with the audience
13:10 - 13:15	***Close (5 minutes)***

Creating a community

BoF website

parallel-code-qa.github.io/sc19-bof

Join the BoF mailing list to receive relevant updates and discussions

groups.io/g/parallel-code-qa-sc19-bof

Creating a community

BoF website

parallel-code-qa.github.io/sc19-bof

Join the BoF mailing list to receive relevant updates and discussions

groups.io/g/parallel-code-qa-sc19-bof

SC19 BoF
Quality Assurance and Coding Standards for Parallel Software

parallel-code-qa.github.io/sc19-bof

Manuel Arenaz
manuel.arenaz@appentra.com

appentra
make code parallel

Julian Miller
miller@itc.rwth-aachen.de

IT Center | RWTH AACHEN UNIVERSITY

SC19
Denver, CO | hpc is now.

Unión Europea
Fondo Europeo
de Desarrollo Regional
"Una manera de hacer Europa"

BoF

Quality Assurance and Coding Standards for Parallel Software

Nov , 21 - 12:15 pm
Room 505

Panelists

David Bernholdt (ORNL, US)

Saber Feki (KAUST, Saudi Arabia)

Dirk Pleiter(Juelich Supercomputing Center, Germany)

Chunhua "Leo" Liao (LLNL, US)

Robert Schiela (Software Engineering Institute, Carnegie Mellon University, US)

Khaled Elamrawi (Brightskies)

Manuel Arenaz (Appentra Solutions, Spain)

Julian Miller: RWTH Aachen University, Germany)