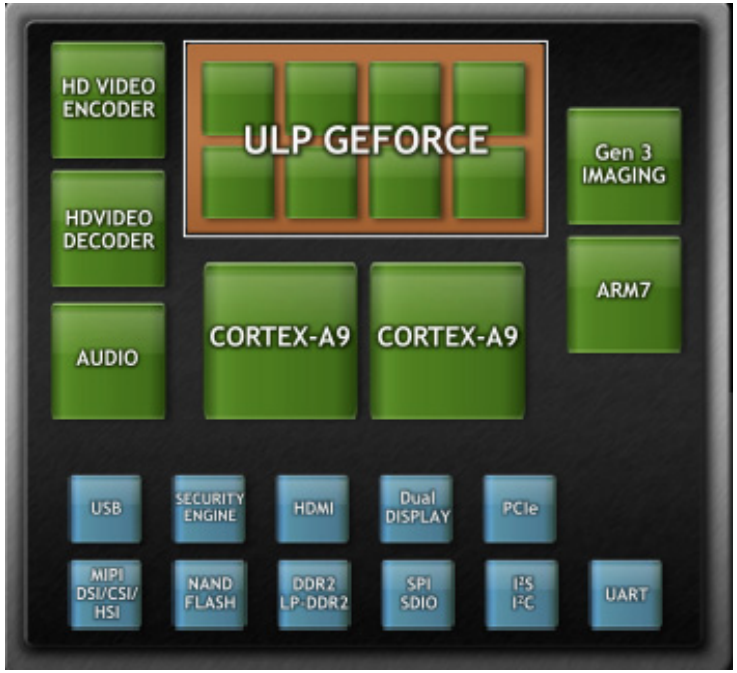
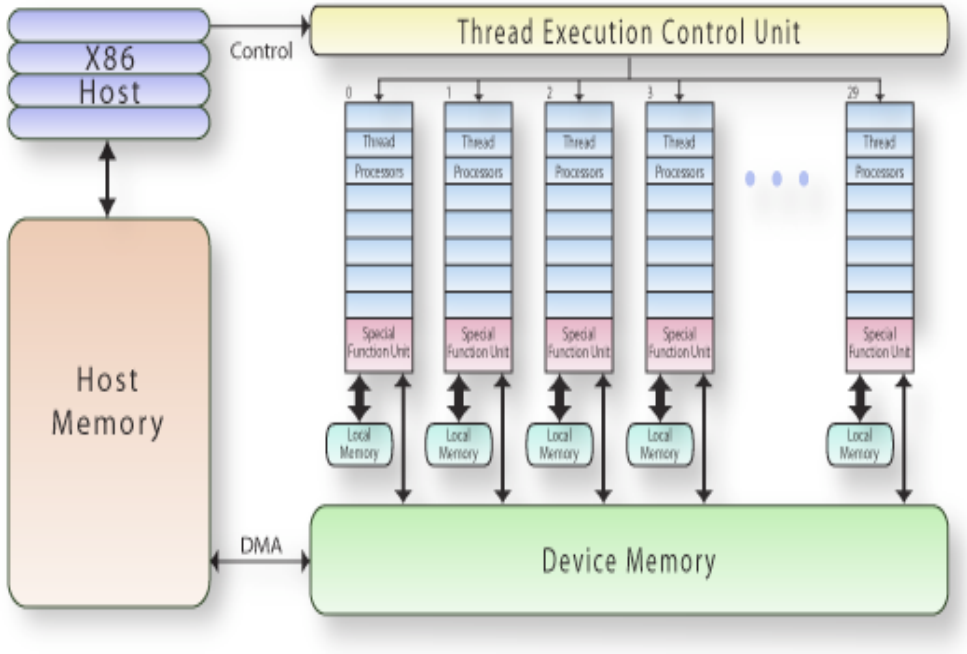
Performance Bugs on real world heterogeneous architecture

Tsung Tai Yeh

Purdue University

Heterogeneous GPU architecture





Integrated GPU

+ Exist in Mobile, NB
+ Low power (ARM+Terga)
+ Share global Memory

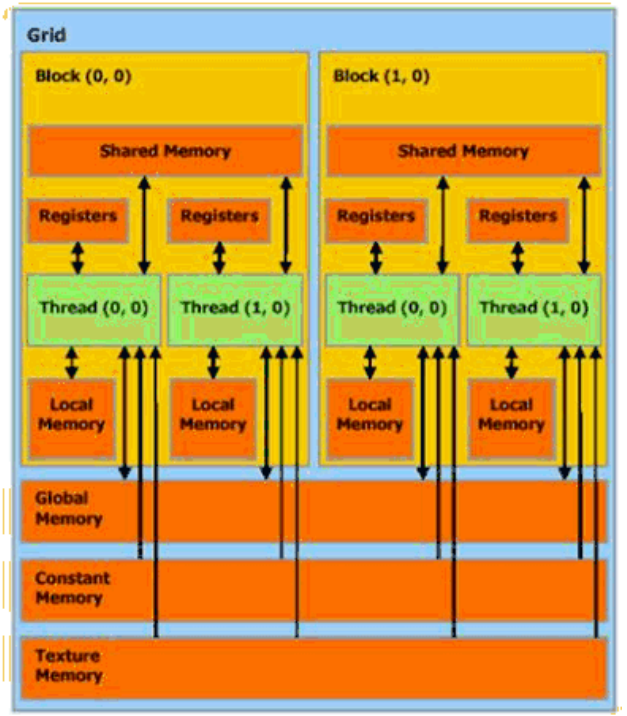
Discreted GPU

+ Exist in Laptop, Servers
+ High performance
+ Specified global Memory

Heterogeneous Architecture is everywhere

Heterogeneous Architecture Programming

- Programming Language
  - CUDA
  - OpenCL
  - OpenACC
- Programming Model
  - Interact with CPU-GPU
  - Hierarchical Thread Architecture
  - Multiple Memories (Shared, context, global memory)



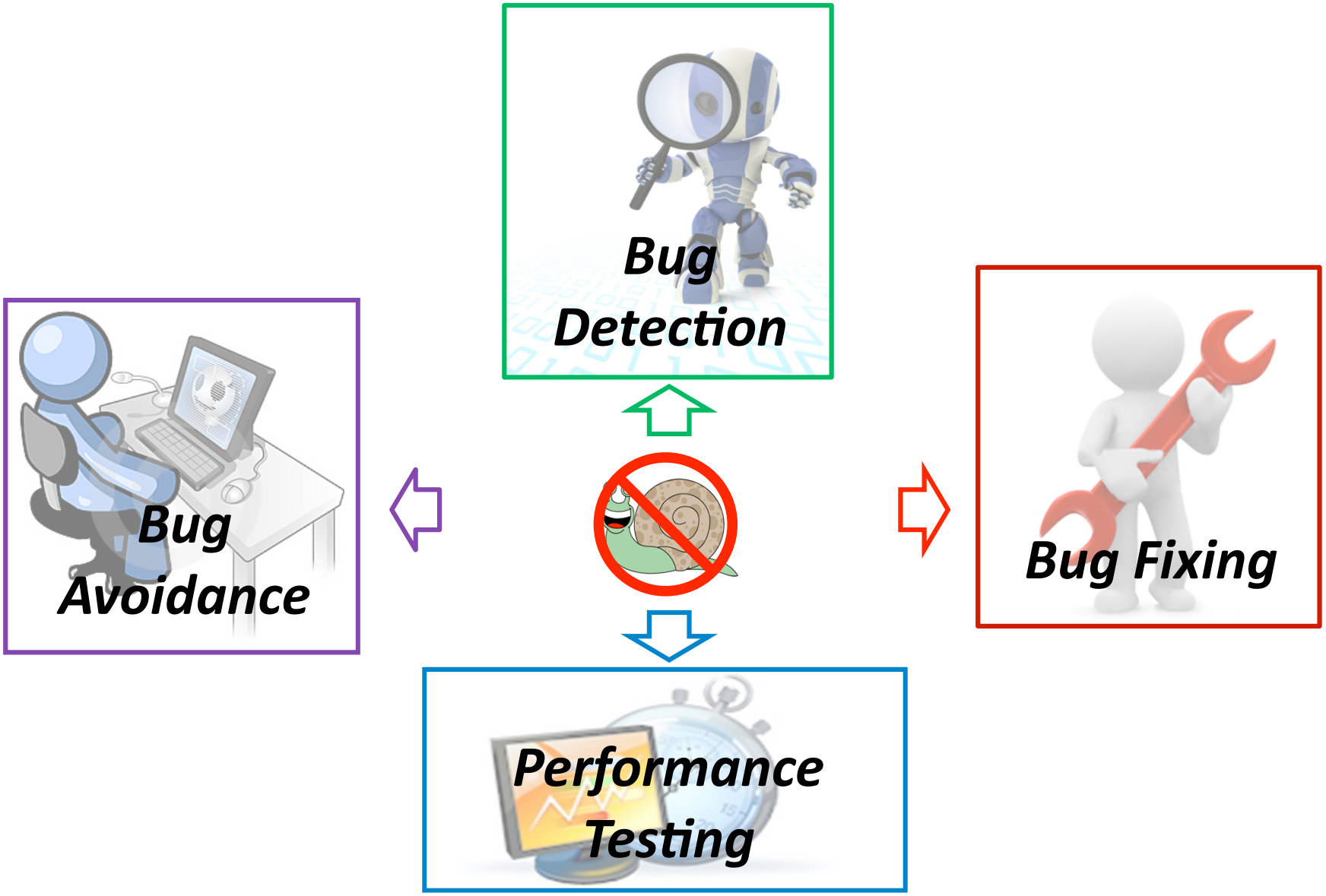
What's critical issue on heterogeneous architecture

- Power, and performance

Specifications	CPU (Intel i7)	GPU (Geforce GTX 690)
# of cores	4	3072
Clock speed	2.5 GHz	975MHz
Memory Bandwidth		384GB/sec
Power consumption	140-160W	300W

Source: Nvidia, Intel

How to fight Performance Bugs



What is performance bugs

- Performance bugs
  - The defects where relatively simple source-code changes can significantly speed up software, while preserving functionality.
- Software efficiency is increasing important
  - Hardware is not getting faster (per-core)
  - Software is getting more complex
  - Energy saving is getting more urgent

Performance bugs and traditional bugs

- Root-cause of bugs
  - Similar with traditional bugs, since they are all related to usage rules of functions/APIs
- Bug-fixing
  - Developers cannot fight performance bugs by themselves, since they cannot predict future workload or code changes to avoid bugs
- Long life time of performance bugs
  - The 36 Mozilla bugs took 935 days on average to get discovered, and another 140 days on average to be fixed, compared to the functional bugs, functional bugs took 252 days on average to be discovered, and 117 days to be fixed.

What is the challenge of performance bugs

- Not compiler-associated
  - These defects cannot be optimized away by compilers.
- Drawbacks of testing techniques
  - Traditional testing method allows the most performance bugs to escape.
  - Performance bugs need inputs with special features to manifest.
  - Performance bugs need large-scale inputs to manifest in a perceivable way.
  - Non fail-stop symptoms.

How performance bugs arise from?

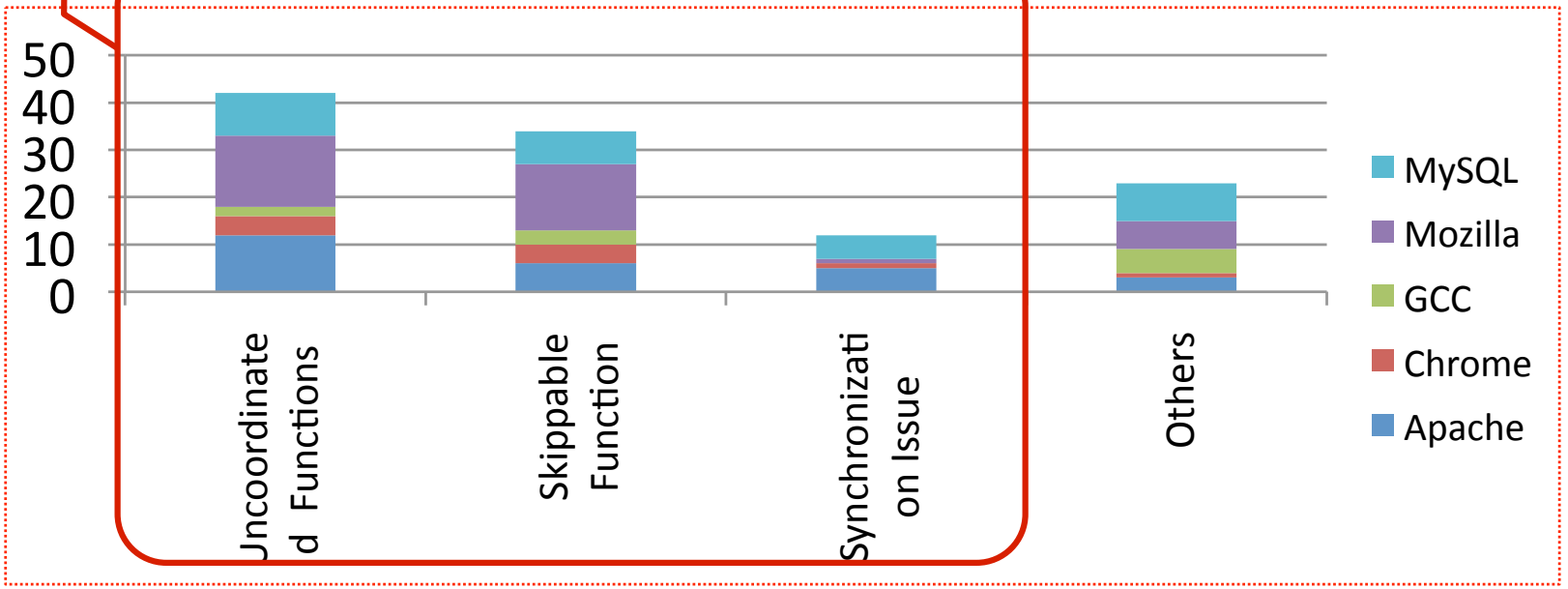
- Misuse inefficient function-call combinations
- Skippable function
  - Conduct unnecessary work given the calling context
- Synchronization issues
  - Unnecessary synchronization intensifies thread competition
- Others
  - Wrong data structures, hardware architecture issues, high-level design/algorithm issues

Root Causes of Performance Bugs

Implication: Future bug detection research should focus on these common root causes.

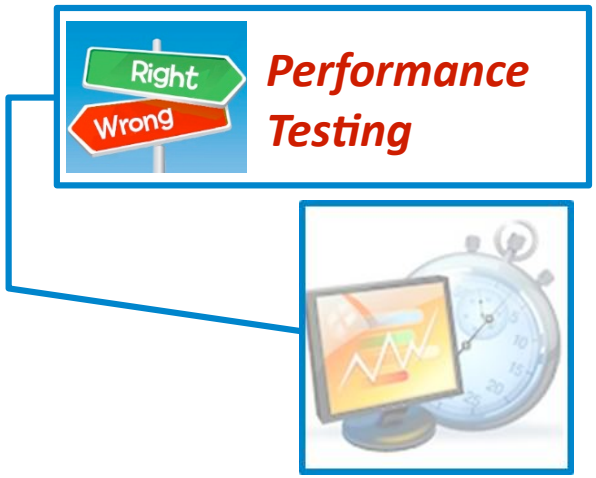
Performance Bug Detection

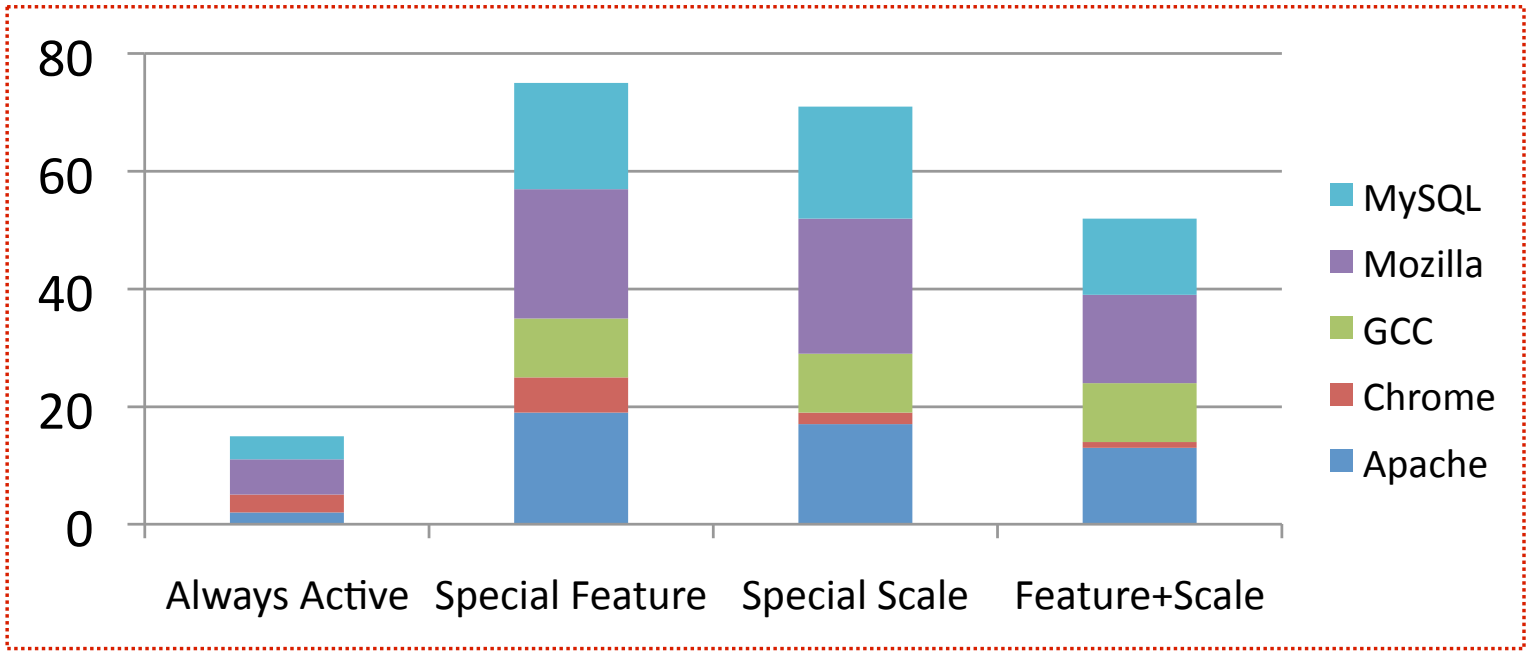
Dominating



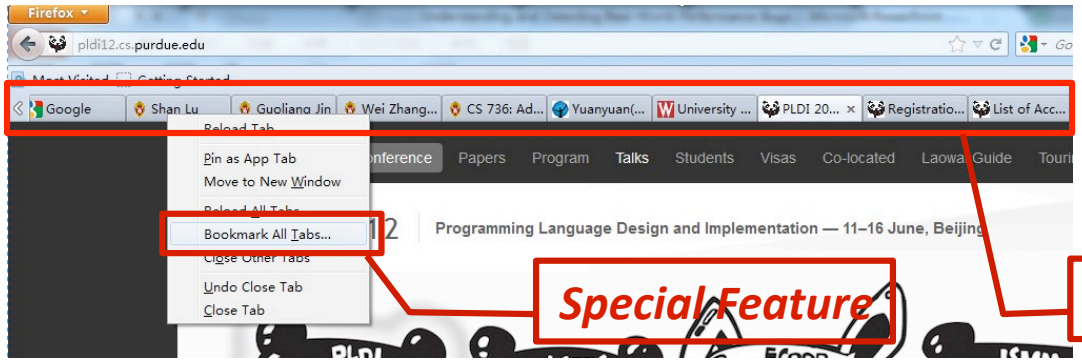
How Performance Bugs Manifest

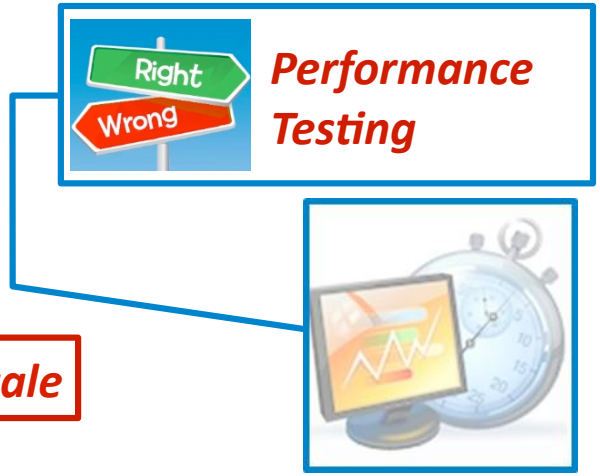
Implication: New input generation tools are needed.

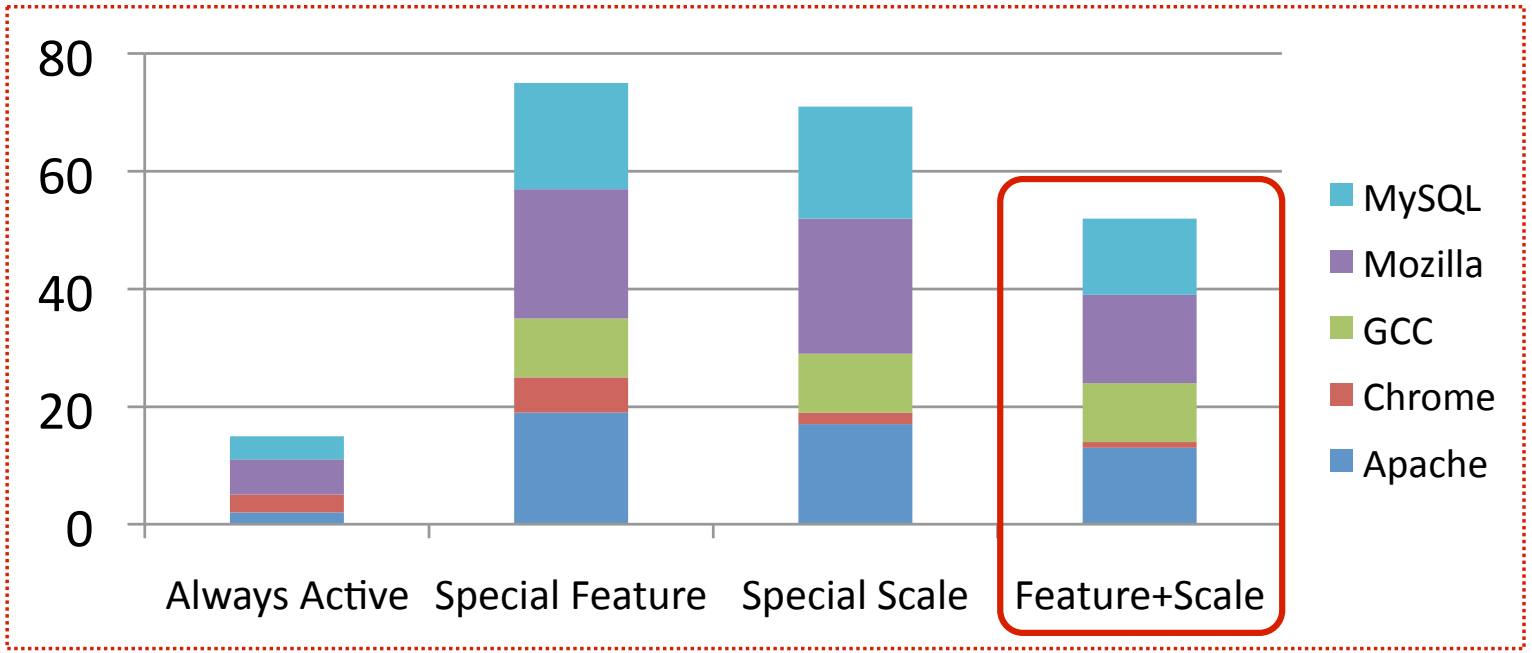




How Performance Bugs Manifest

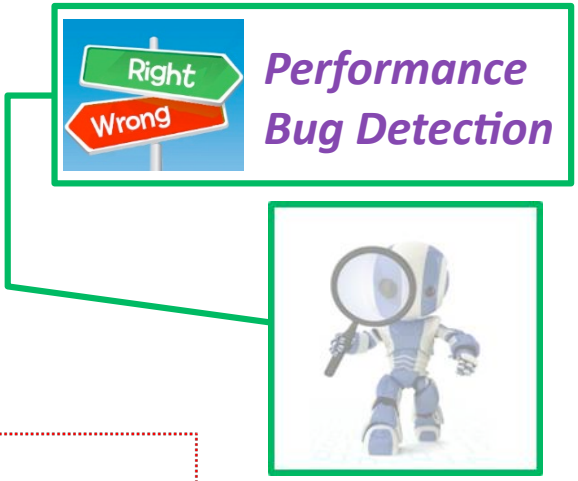


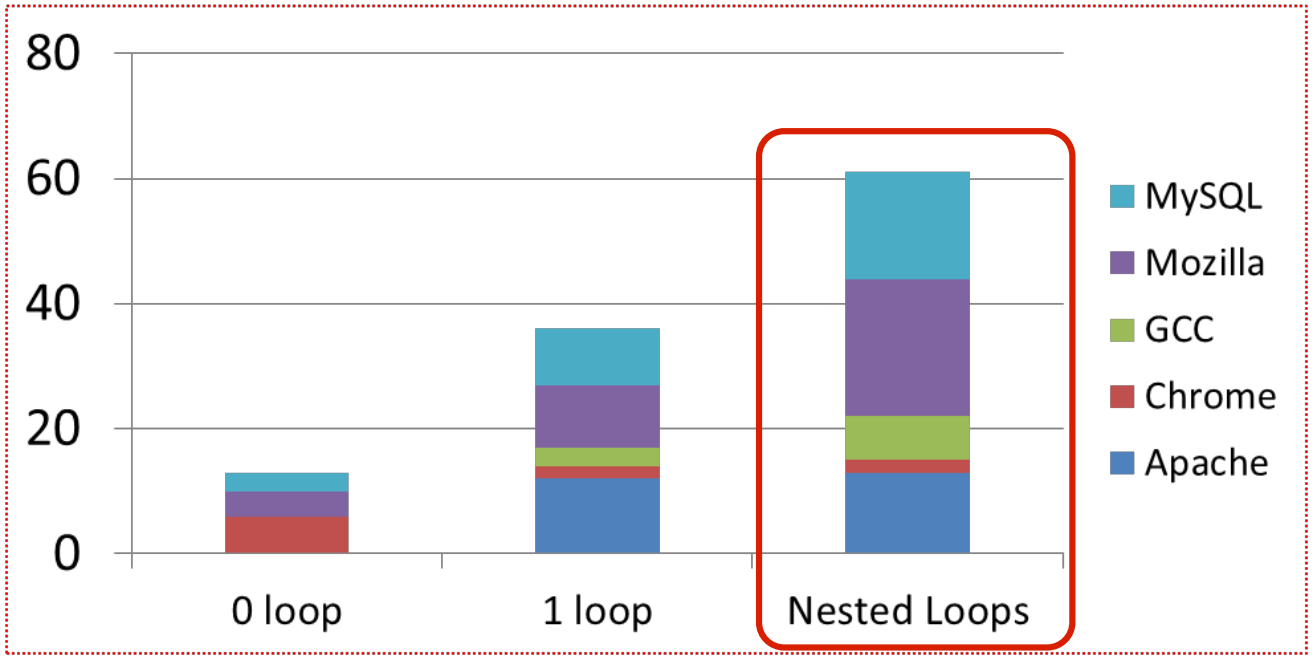




Locations of Performance Bugs

Implication: Detecting inefficiency in nested loops is critical.

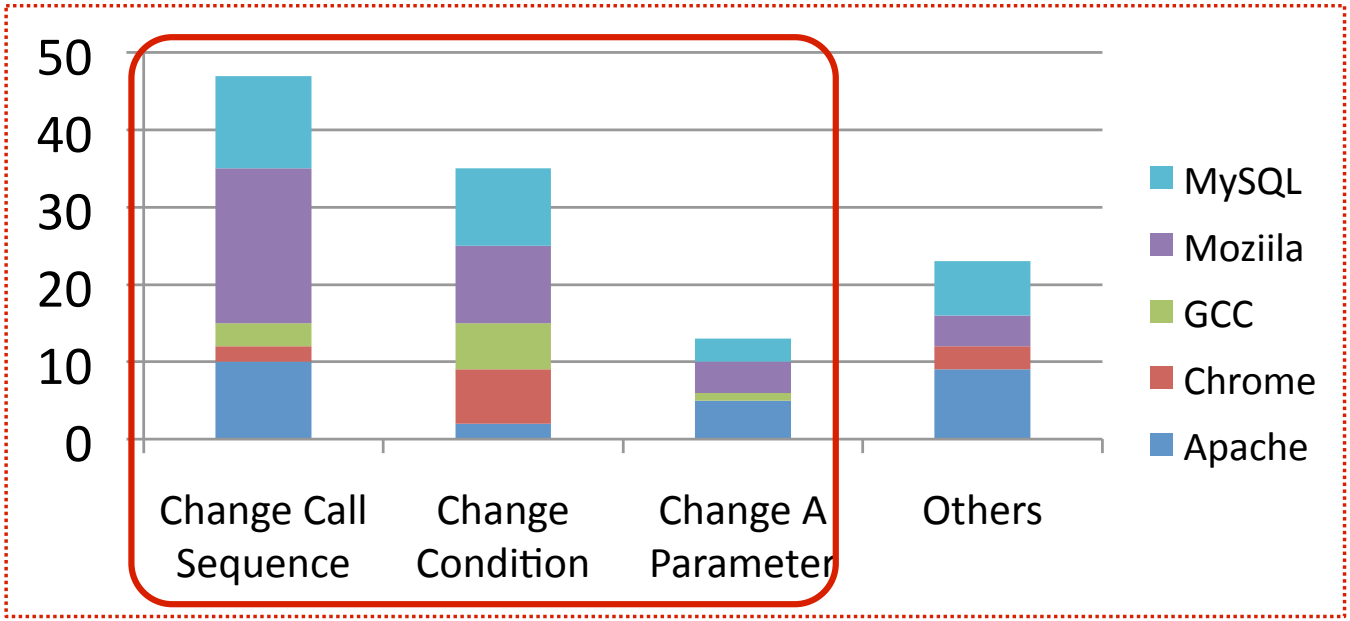




How Performance Bugs are Fixed

- Patch sizes are small
  - 42 patches are no larger than 5 LOC
  - Median patch size = 8 lines of codes

➔Fixing perf. bugs does not hurt readability



Performance bugs on GPU

- Global memory data access patterns
  - Adjacent, row-based, shared data accesses
- Block dimensions in thread hierarchy
  - No data reuse through shared memory
- Code Portability
  - Different GPU necessitate different performance consideration
- Function specialization
- Floating-point number computations

Performance bugs and Reliability

- Performance and energy
  - Performance ↑, and energy cost ↓
  - Energy = power x time (dynamic power + leakage power) x time
- Energy and temperature (heat)
  - Energy ↑, and temperature ↑
- Heat and reliability
  - Heat ↑, and reliability ↓

The future works

- Performance bugs recognition
  - Cause by Heterogeneous architecture
  - Cause by Parallel programming
  - What these bugs look like?
  - What difference between traditional bugs?
  - What the root cause of these bugs?
- Performance bugs detection
  - How to detect these bugs?
  - How different compare with traditional bug detection techniques

The future works

- Bug avoidance
  - How to fix them?
  - Automatic or manual?
- Modeling performance bugs and system reliability
  - How performance bugs affect system, and hardware reliability?
  - Energy-aware, reliability-aware programming

Conclusion

- Performance bugs research
  - Performance bugs are getting to be critical in computing
  - The performance bugs analysis research is in infant stage
  - The complexity of software cause performance bugs are difficult to discover
- Heterogeneous architecture programming
  - Portability
  - Power, performance

Reference

- [1] Guoliang Jin, **Linhai Song**, Xiaoming Shi, Joel Scherpelz, Shan Lu, Understanding and Detecting Real-World Performance Bugs , PLDI, 2012
- [2] Y. Yang, P. Xiang, M. Mantor, and H. Zhou, "Fixing Performance Bugs: An Empirical Study of Open-Source GPGPU Programs", ICPP, 2012
- [3] Li, Guodong and Li, Peng and Sawaya, Geof and Gopalakrishnan, Ganesh and Ghosh, Indradeep and Rajan, Sreeranga P., GKLEE: concolic verification and test generation for GPUs, PPoPP 2012
- [4] Zheng, Mai and Ravi, Vignesh T. and Qin, Feng and Agrawal, Gagan, GRace: a low-overhead mechanism for detecting data races in GPU programs, PPoPP 2011
- [5] Nvidia, www.nvidia.com