



Exploring adaptive program behavior

Bonnichsen, Lars Frydendal; Probst, Christian W.

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):

Bonnichsen, L. F., & Probst, C. W. (2014). Exploring adaptive program behavior. Poster session presented at International Symposium on Code Generation and Optimization, CGO 2014, Orlando, Florida, United States.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Motivation

- ▶ Modern computer systems are ever more complex.
 - ▶ Their bottlenecks can change at run time.
 - ▶ The dynamic nature makes it more difficult to make good optimization decisions.
- ▶ Adaptive behavior has been suggested as a solution.
 - ▶ Decisions are made at runtime by switching between program **versions**.

Version: implementation of part of a program.

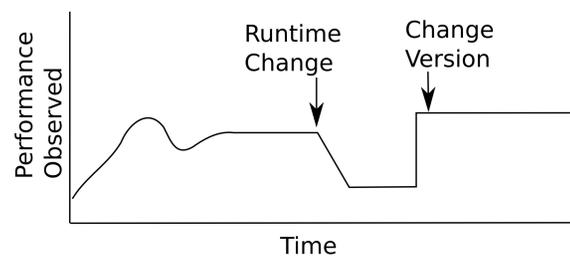


Figure : Adapting behavior based on observed performance.

- ▶ Adaptive behavior grants the optimization process more information, but comes at a cost.

Gained information	Costs
Run time bottlenecks	Monitoring and indirections
Input data	Switching versions
Problem size	Finding versions

- ▶ We need to explore the design space of adaptive behavior, to strike a good balance.

Approach

- ▶ There are different costs and gains for different forms of adaptive behavior:
 - ▶ For instance, recompilation, dispatchers, and iterative compilation.
- ▶ We want to integrate and evaluate different forms of adaptive behavior.
- ▶ Our strategy has 3 steps:
 1. Design an extensible adaptivity framework.
 2. Expose adaptivity uniformly.
 3. Use the extensible nature of the framework to support a wide range of adaptive behavior.

Adaptivity framework

- ▶ We will optimize based on feedback, both at run time and compile time.
- ▶ We will use a uniform design to explore adaptive program behavior, as outlined in below:

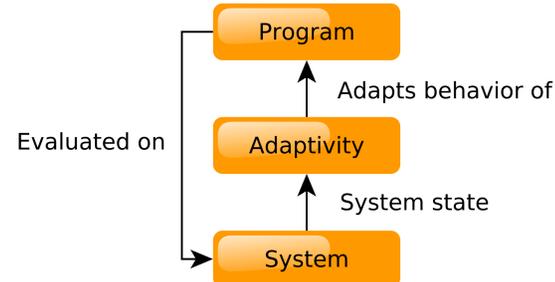


Figure : Adaptivity as a feedback loop

- ▶ Programs are iteratively refined based on feedback from the system.

Adaptivity operation

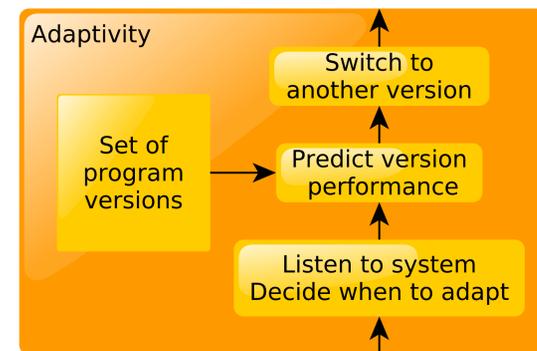


Figure : Detailed view of adaptivity

- ▶ The adaptivity block decides when to adapt, selects a version to use, and switches to it.
- ▶ The actions form a wide design space.

Design space exploration

- ▶ We characterize the adaptivities design space by 4 factors.

Adaptivity Action: how to adapt. **Adaptivity Trigger:** causes for adapting.

Performance Prediction: predict versions performance. **Version Sources:** sources of alternatives.

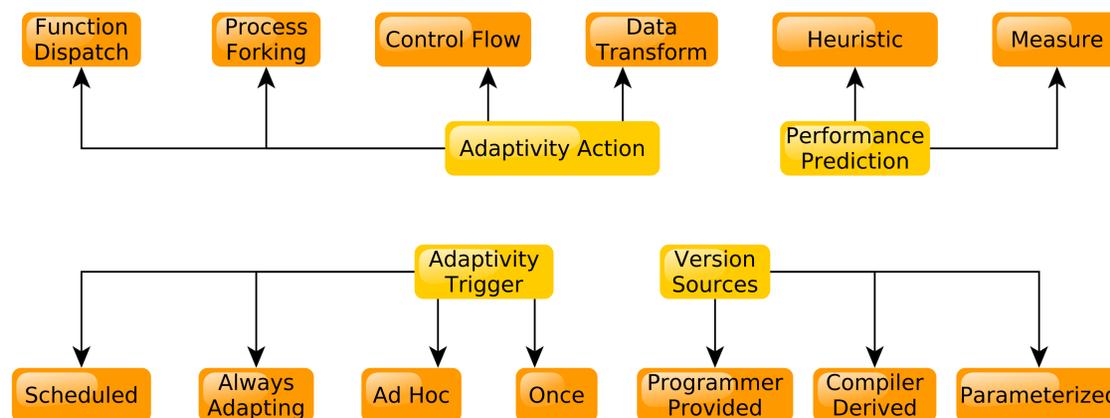


Figure : Adaptivity design space

- ▶ The design space can encode many existing forms of adaptivity, such as:
 - ▶ Iterative compilation: (Process Forking, Once, Measure, Compiler Derived)
 - ▶ Auto-tuning: (Control flow, Once, Measure, Programmer Provided and Parameterized)
- ▶ We have a proof-of-concept adaptivity framework that would be encoded as: (Function Dispatch, Scheduled, Measure and Heuristic, Programmer Provided)
- ▶ We will continuously expand the framework to explore the design space.

Contributions

- ▶ Designed an extensible adaptivity framework.
- ▶ Implemented a function level adaptivity framework.
- ▶ Outlined how the framework can be expanded to the whole design space.
- ▶ Defined criterias to evaluate adaptive program behavior.

Future work

- ▶ Using the outlined structure, we will integrate different forms of adaptive behavior to determine.
 1. How much can be gained from the adaptive behavior. (~ 5 – 16% [1])
 2. The cost of switching versions.
 3. The overhead of supporting switches. (Best case ~ 2% [3])
 4. How should versions be selected.

At a high level, we want to determine when the benefits of adaptive behavior outweigh the disadvantages.

Related Work

- [1] Voss *et al.* High-Level Adaptive Program Optimization with ADAPT. In *PPOPP'01*.
 [2] Fursin *et al.* A Practical Method For Quickly Evaluating Program Optimizations. In *HiPEAC'05*.
 [3] Mars *et al.* Scenario Based Optimization: A Framework for Statically Enabling Online Optimizations. In *CGO'09*.

Conclusion

- ▶ Adaptive behavior allows for further optimization, but it comes at a cost.
- ▶ We need to determine when adaptive program behavior is worth the cost.
- ▶ We have a plan how to integrate different forms of adaptivity.