Cray HPCS Productivity Features

Joint NERSC/OLCF/NICS

Cray XT5 Workshop

February 1-3, 2010

Margaret Cahir
Cray, Inc.

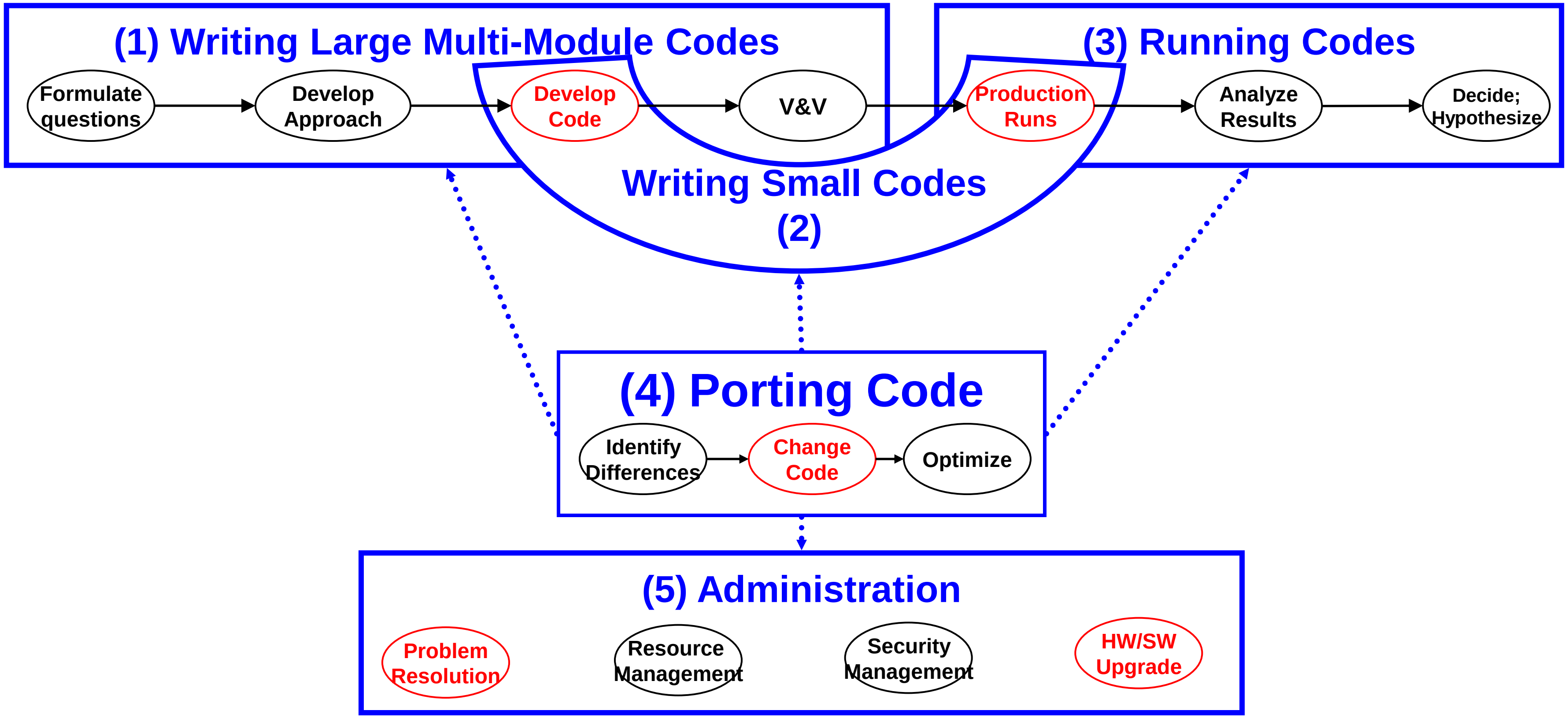
Agenda

- Background on the Productivity Efforts
- 2 Productivity Tools/Features
  - ATP (Abnormal Termination Processing)
  - APA (Automatic Profiling Analysis)
- Assessing Productivity Improvements

Productivity Background

- ***The problem:*** Large-scale scientific computers are getting larger and faster, but also more complex and more difficult to use
  - Complexity is especially challenging to new users
- HPCS Phase III Program specifically calls for improvements in developer productivity
  - This is completely separate from hardware performance improvements
  - Embodied in a set of 5 workflows. Developer productivity comes into play in 3 of them:
    - Writing large (multi-module) codes
    - Writing small codes
    - Porting codes

Level 1 Functional Workflows



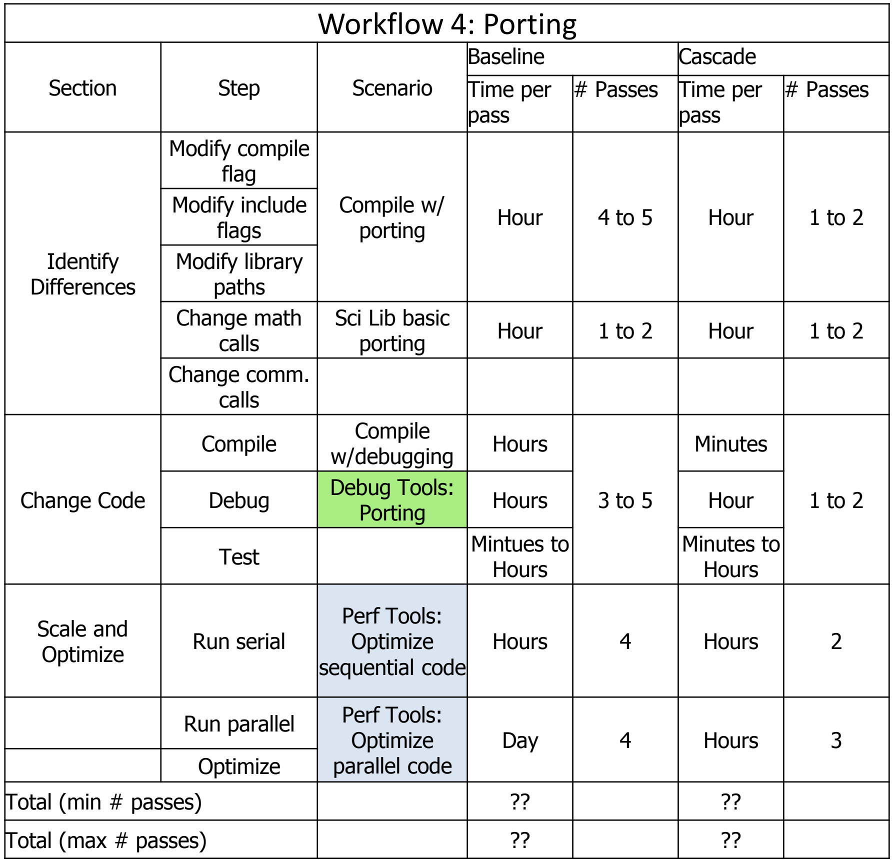
- **Workflows comprise many steps; many overlapping**
- **Item in red represent areas with highest HPC specific interest**

Productivity Feature Work

- Cray is implementing a variety of new software and hardware features aimed at improving productivity
  - System Administration
    - identifying problems
    - upgrading system software
  - Writing new codes
    - Chapel language
      - “global-view” language, designed for parallel programming
      - See chapel.cray.com for more information
  - Compiling, Optimization and Debugging
    - Many features…. Luiz’s talk will cover this
    - Includes ATP and APA

Feature Assessments and Workflows

- Assess individual features or tools for their contribution to improving developer productivity
  - Compare how much time/effort when using the tool or feature vs. what effort was involved in the 2002 timeframe
- Will apply those improvements towards the workflows
- Starting with evaluations of 2 features:
  - ATP – a debug tool
  - APA – a feature of the performance analysis tool (CrayPat)



Workflow 4: Porting

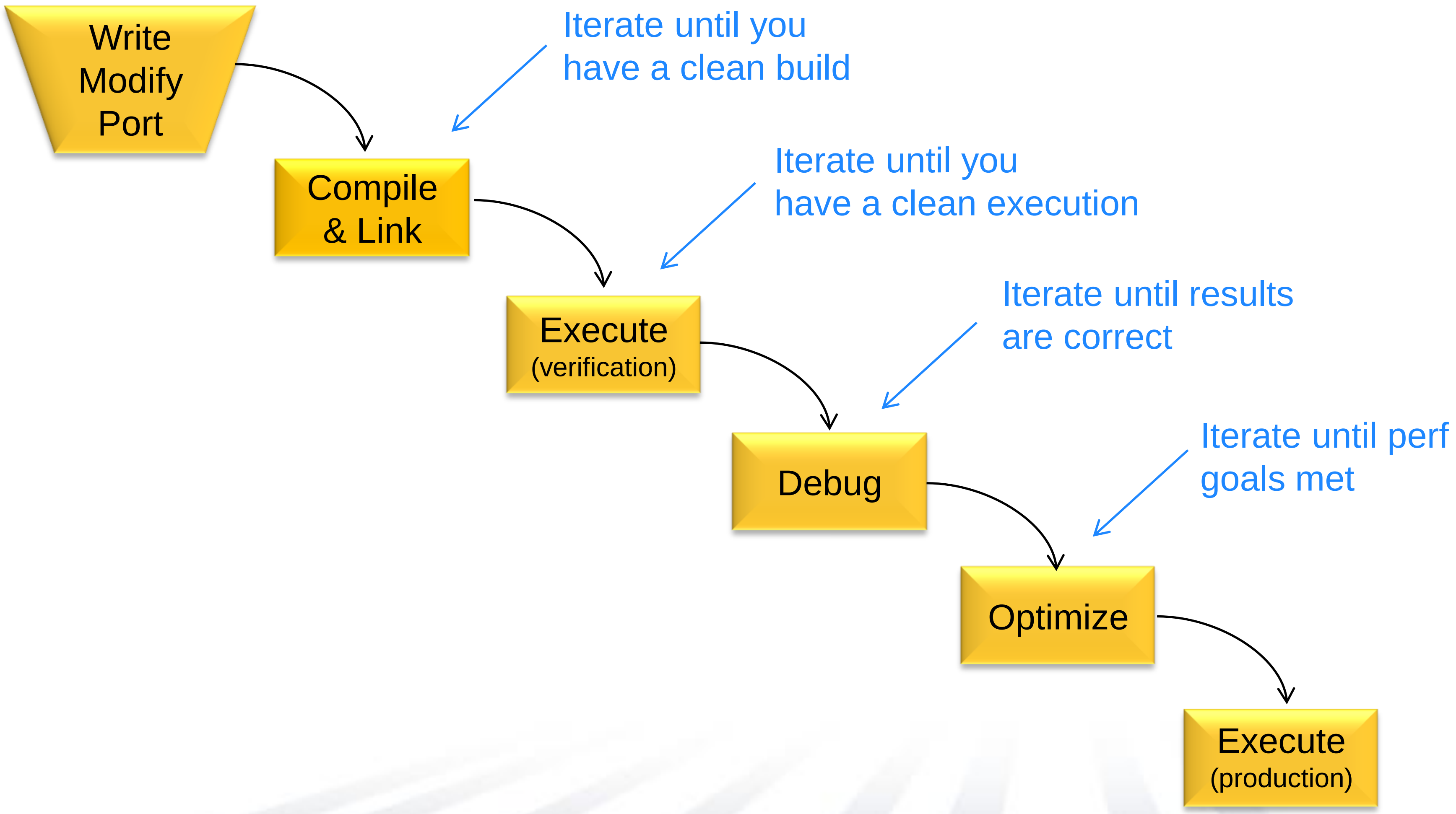
Section	Step	Scenario	Baseline		Cascade	
			Time per pass	# Passes	Time per pass	# Passes
Identify Differences	Modify compile flag	Compile w/ porting	Hour	4 to 5	Hour	1 to 2
	Modify include flags					
	Modify library paths					
	Change math calls	Sci Lib basic porting	Hour	1 to 2	Hour	1 to 2
	Change comm. calls					
Change Code	Compile	Compile w/debugging	Hours	3 to 5	Minutes	1 to 2
	Debug	Debug Tools: Porting	Hours		Hour	
	Test		Mintues to Hours		Minutes to Hours	
Scale and Optimize	Run serial	Perf Tools: Optimize sequential code	Hours	4	Hours	2
	Run parallel	Perf Tools: Optimize parallel code	Day	4	Hours	3
	Optimize					
Total (min # passes)			??		??	
Total (max # passes)			??		??	

Simplified Example of a Porting Workflow

ATP – Abnormal Termination Processing

- ***The Problem:*** When a parallel application dies, it is next to impossible to examine all the core files and backtraces
  - Core files
    - A single core file is usually not enough to debug
    - Sufficient storage for all core files is a problem
  - Backtraces
    - A single backtrace is usually not enough
    - The backtrace produced might not be from the process that first failed
    - Today’s systems produce one or none
- ATP produces a single merged stack trace or reduced set of core files. ***The benefits:***
  - Easy to navigate the merged stack trace
  - Manageable set of core files
  - Reduced amount of data saved
    - Especially true in the core file situation

Simplified Workflow – Major Steps



Write
Modify
Port

Iterate until you
have a clean build

Compile
& Link

Iterate until you
have a clean execution

Execute
(verification)

Iterate until results
are correct

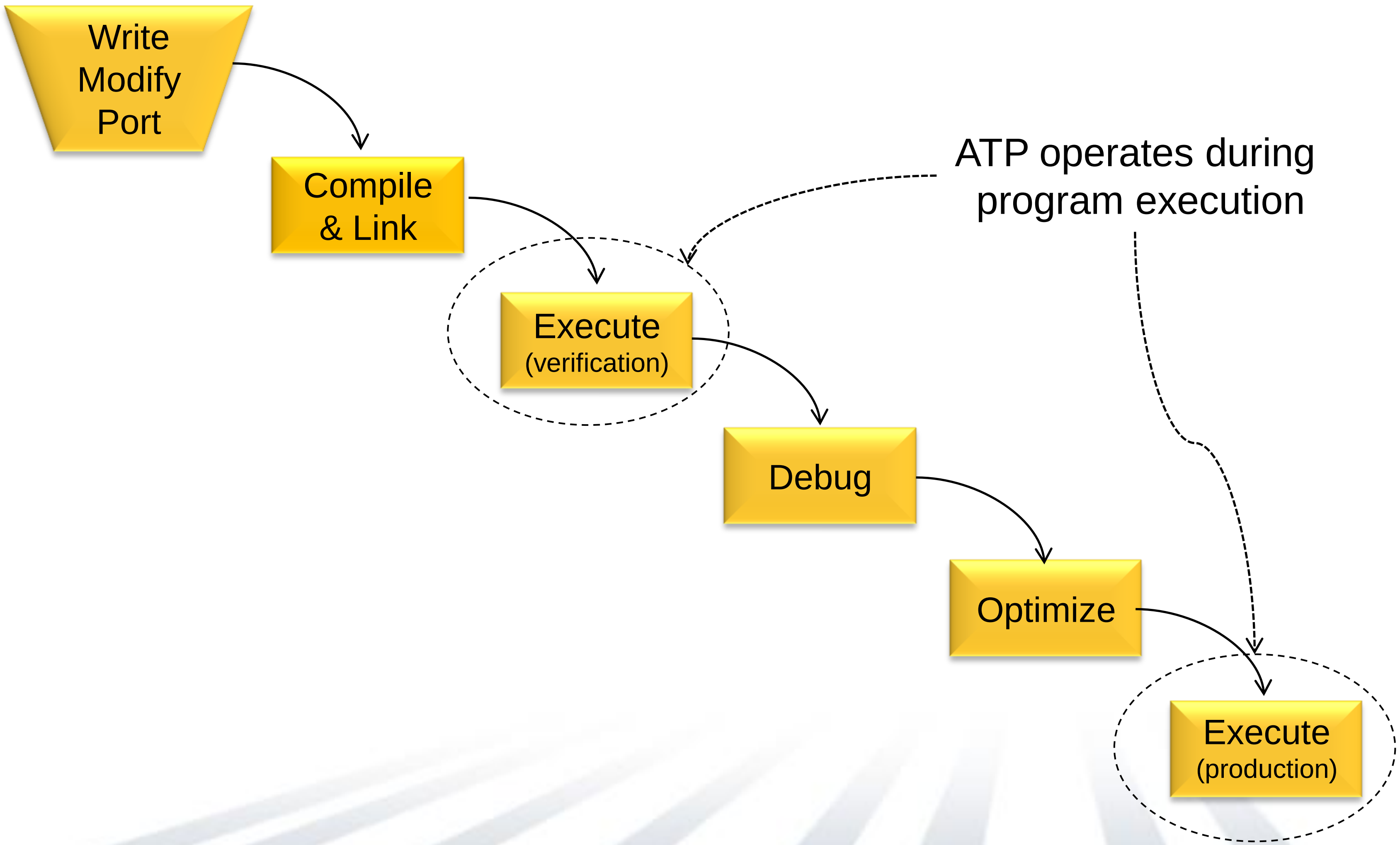
Debug

Iterate until perf
goals met

Optimize

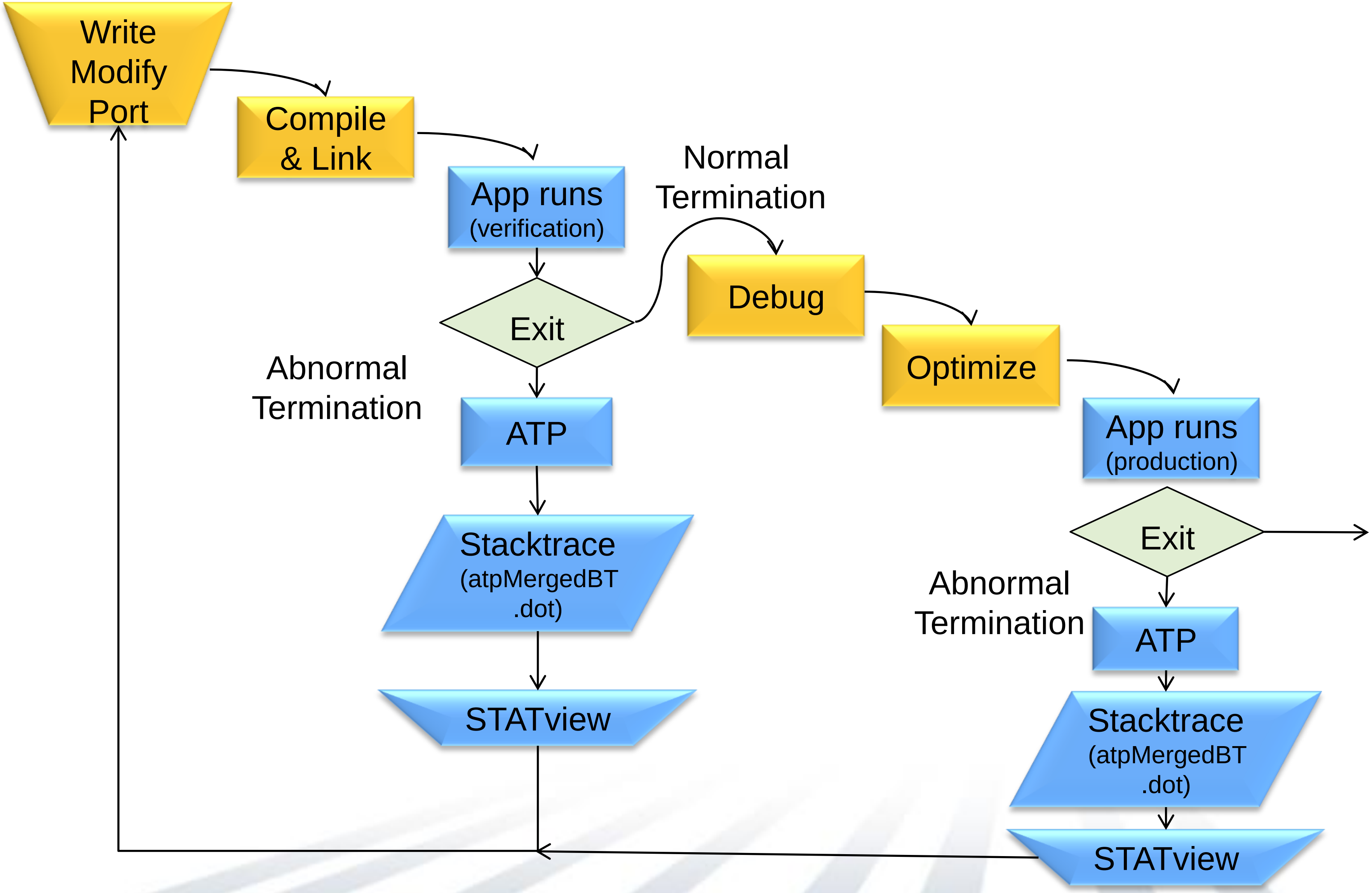
Execute
(production)

Simplified Workflow with ATP



ATP operates during program execution

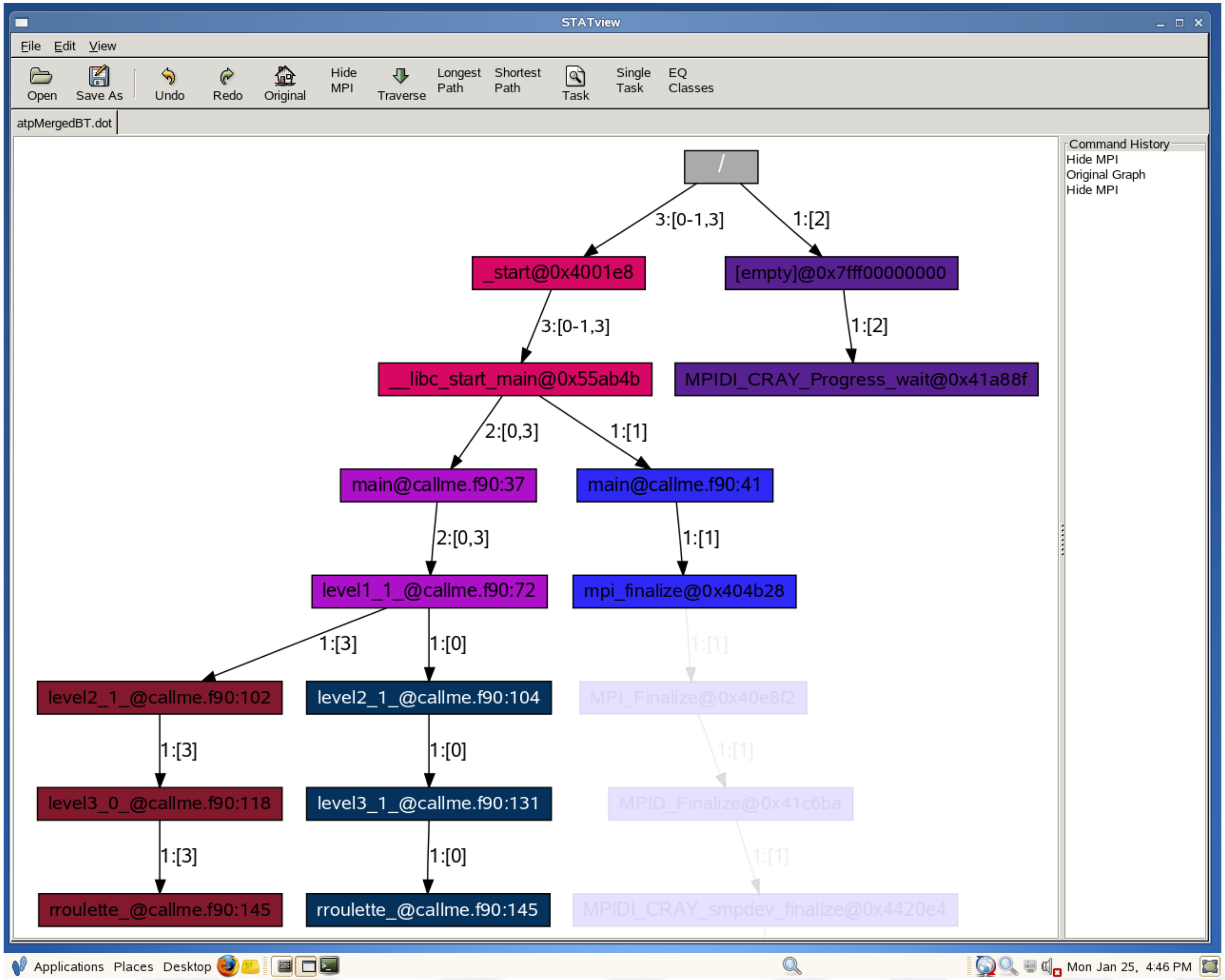
ATP – Abnormal Termination Processing



ATP – How It Works

- ATP signal handler runs within an application. Its job is to catch fatal errors. It handles the following signals:
  - SIGQUIT, SIGILL, SIGTRAP, SIGABRT, SIGFPE, SIGBUS, SIGSEGV, SIGSYS, SIGXCPU, SIGXFSZ
  - Setting the environment variables MPICH_ABORT_ON_ERROR and SHMEM_ABORT_ON_ERROR will cause a signal to be thrown and captured for MPI and SHMEM fatal errors
- ATP daemon running on the compute node captures signals, starts termination processing
  - Rest of the application processes are notified
  - Generates a stacktrace
  - Creates a file named *.dot
- The *.dot file is viewed with the STATview tool
  - Pre-release of STATview is available on workshop systems

STATview Example



ATP – Future Features

- Automatic invocation of ATP
  - Today users need to insert signal handler
  - With next release of OS, just need to load atp module
- Core file subset
  - Intelligence from stack backtrace help decides which core files to produce
- Hold a dying application in stasis
  - Gives the user an opportunity to attach a debugger to the application
- Send email notification to user that job has failed
- Improved scalability
  - ATP stack backtraces have been produced on applications made up of about 2000 processes
  - Expect to be able to handle applications with 100,000s of processes in the future

ATP – Getting Started

- Get atp_example.tar from the Workshop website

```
$ wget http://www.nersc.gov/projects/workshops/CrayXT/tbd
$ tar -xvf atp_example.tar
```

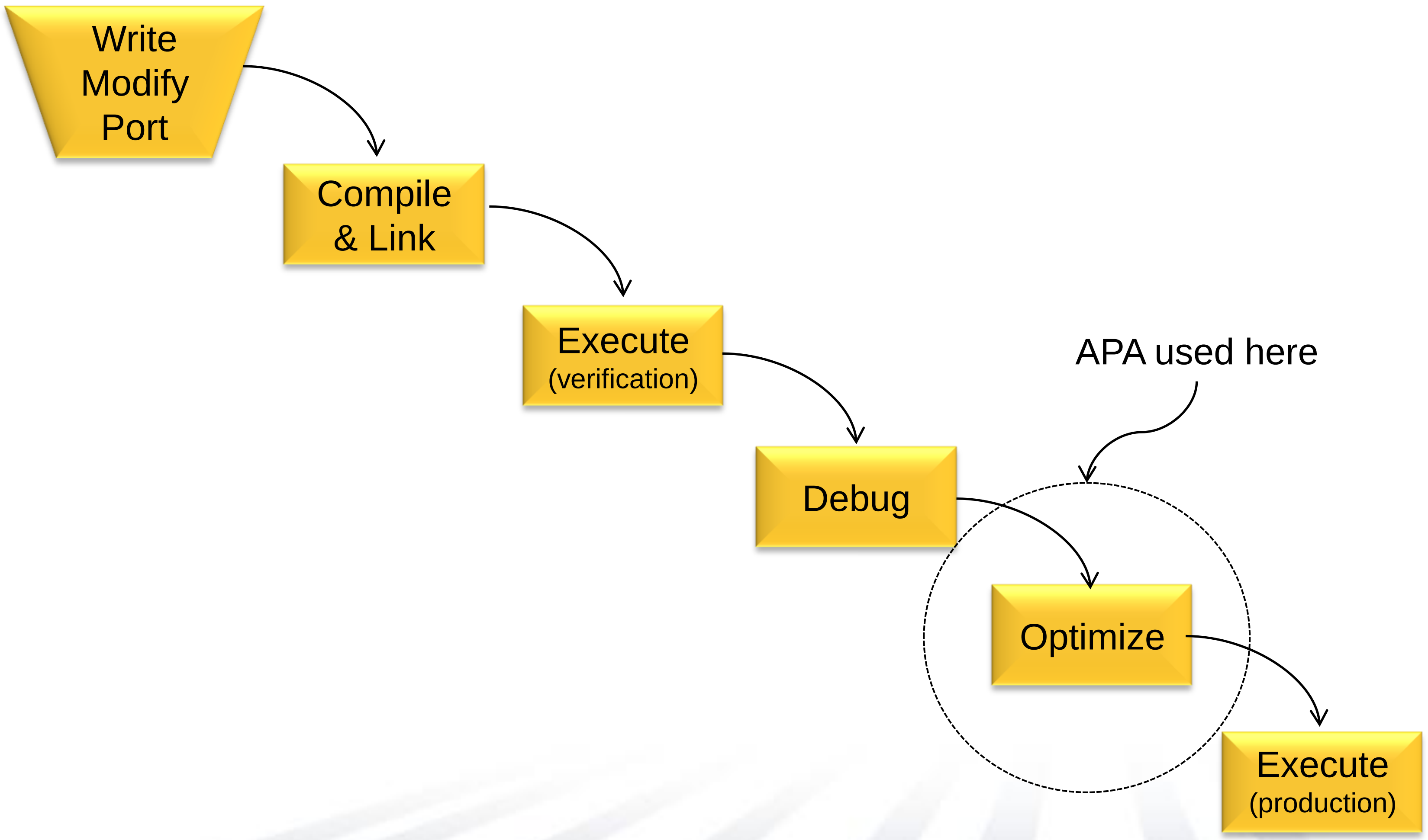
- On a Cray XT with atp installed, type:

```
$ module load xt-atp
$ module load stat
$ man intro_atp
```

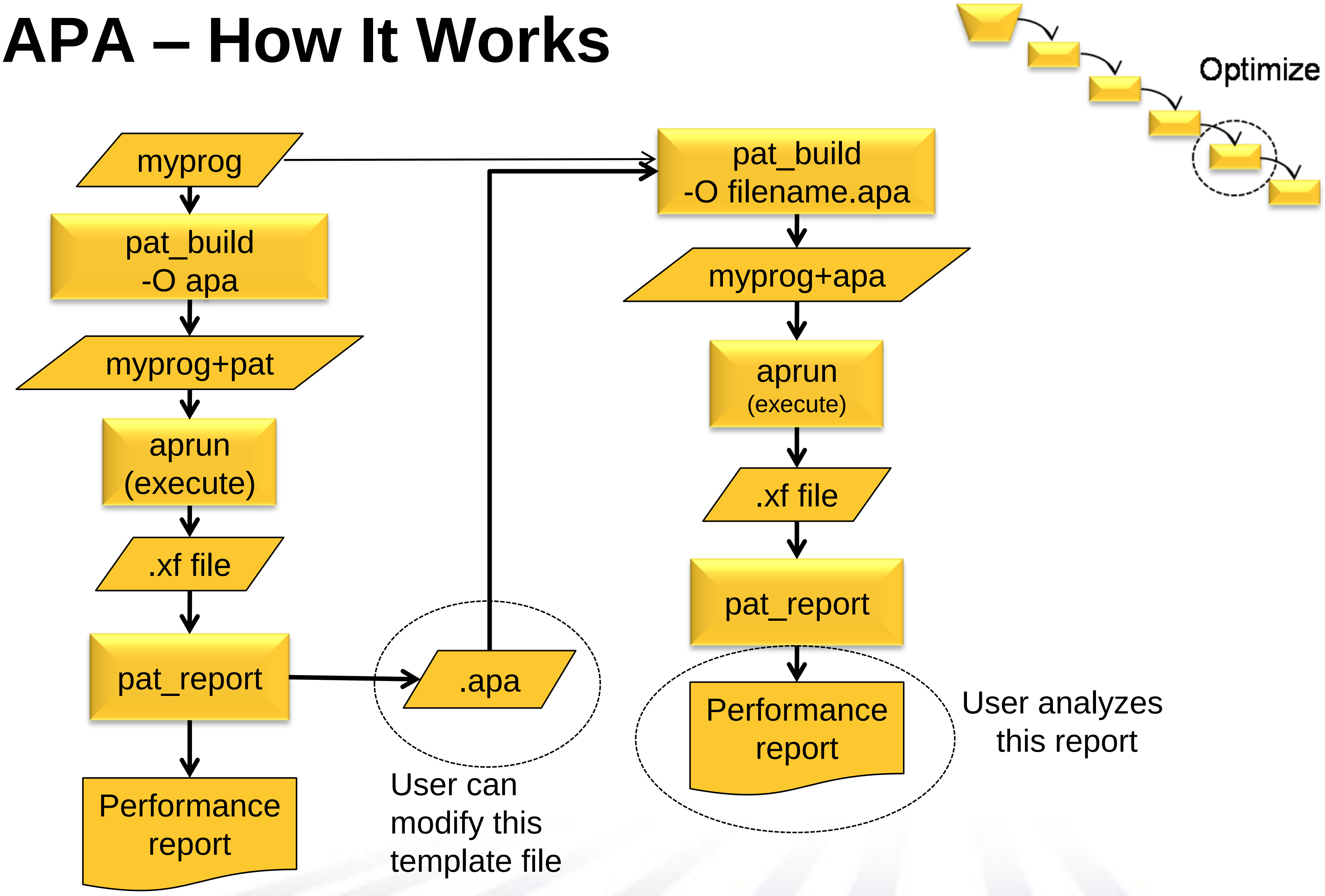
APA – Automatic Profile Analysis

- ***The Problem:*** performance tools have many options and it can be a lot of work to set up options to profile a program with minimum overhead
- APA is an option that automatically creates a template file that can be used to set up a performance profile of the run
- ***The Benefits:***
  - You can quickly and efficiently generate a performance profile
    - Automatically excludes those routines which took a small amount of time to reduce runtime overhead
    - Automatically specifies hardware counter groups
    - Automatically lists which libraries to profile
  - You do not need to wade through pages of documentation in order to do this
  - The template (.apa) file can subsequently be modified to refine the performance data collection
    - Also serves as usage documentation

Simplified Workflow with APA



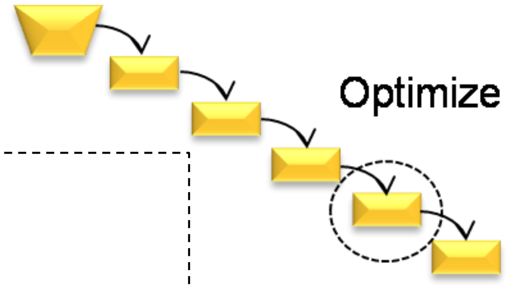
APA – How It Works

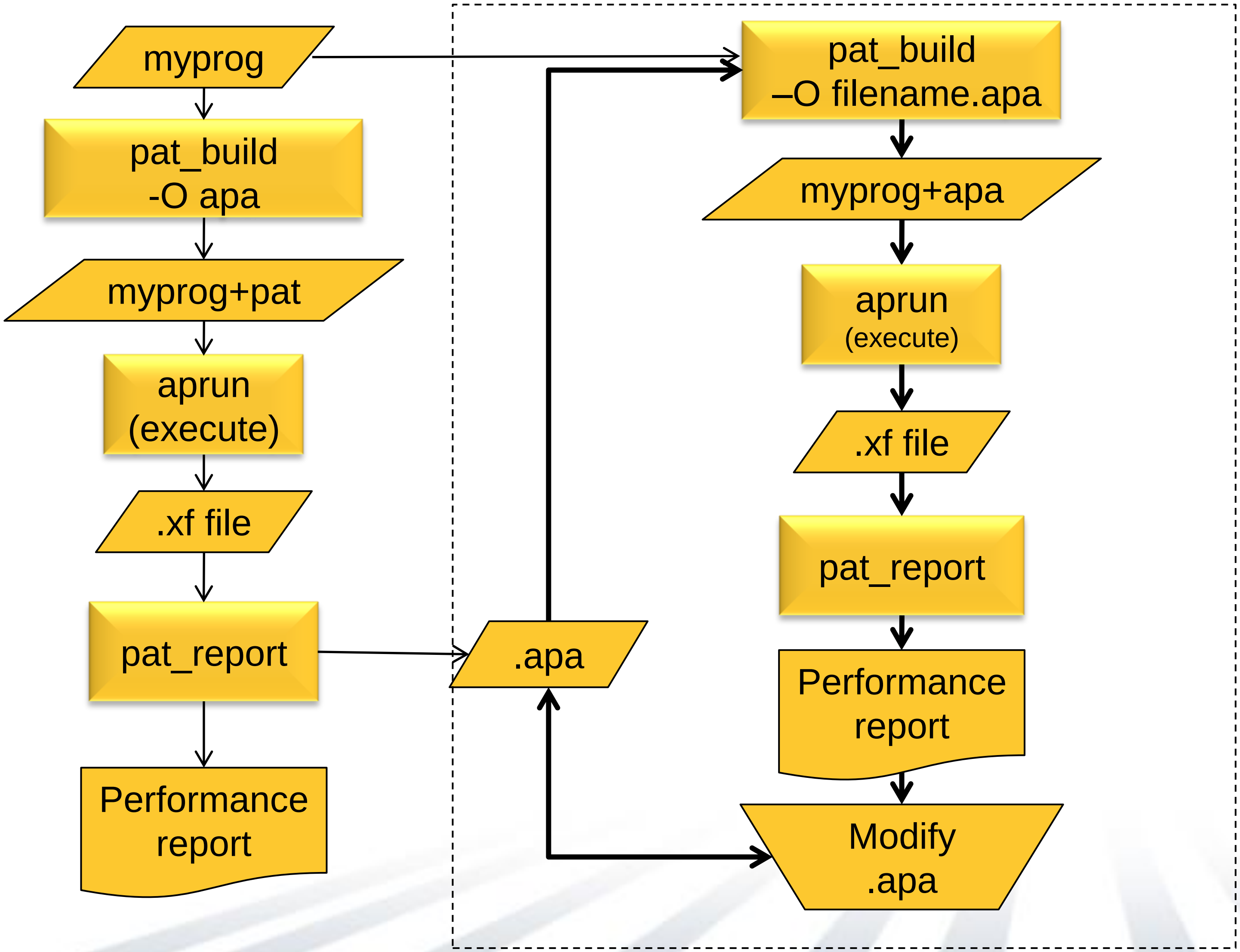


User can modify this template file

User analyzes this report

APA – Subsequent Iterations





APA – How It Works

- User first instruments code with `pat_build -O apa`
  - Straightforward and requires little overhead when running
- User executes the application
  - The information needed to make a profile run is generated and produced in a file with the extension .apa
- Reinstrument the code (using .apa file)
- Rerun the code (produces .xf file)
- Produce the profile report

APA – Getting Started

- Get apa_example.tar from the Workshop website

```
$ wget http://www.nersc.gov/projects/workshops/CrayXT/tbd
$ tar –xvf apa_example.tar
```

- Alternatively:
  - See Section 2.4 *Using Automatic Program Analysis* in the manual *Using Cray Performance Analysis Tools* S-2376-50
  - Available on the docs.cray.com website

- Another alternative:

```
$ module load xt-craypat
$ man intro_craypat
```

Feature Assessments

- Objective is to answer the following questions:
  - *Does this feature help boost the productivity of developers?*
  - ***How much** does it help?*
  - ***How easy** was it to learn how to use the feature?*
- We asking users to try out these features and report back on their experience
- We are providing:
  - Quick, get-started guide for each feature which includes
    - Feature description
    - Feature benefit
    - How to
  - Simple example
    - Includes a shell script which walks through the steps

Feedback

- How and when
  - Fill in provided feedback forms during workshop
  - Talk to us during Hands-on time
  - Contact us via email
    - Margaret Cahir n13671@cray.com
    - Don Mason dmm@cray.com
- Would like to gather initial impressions of new tools and features
  - How easy it was to learn
  - How useful will it be
  - Time spent is of interest