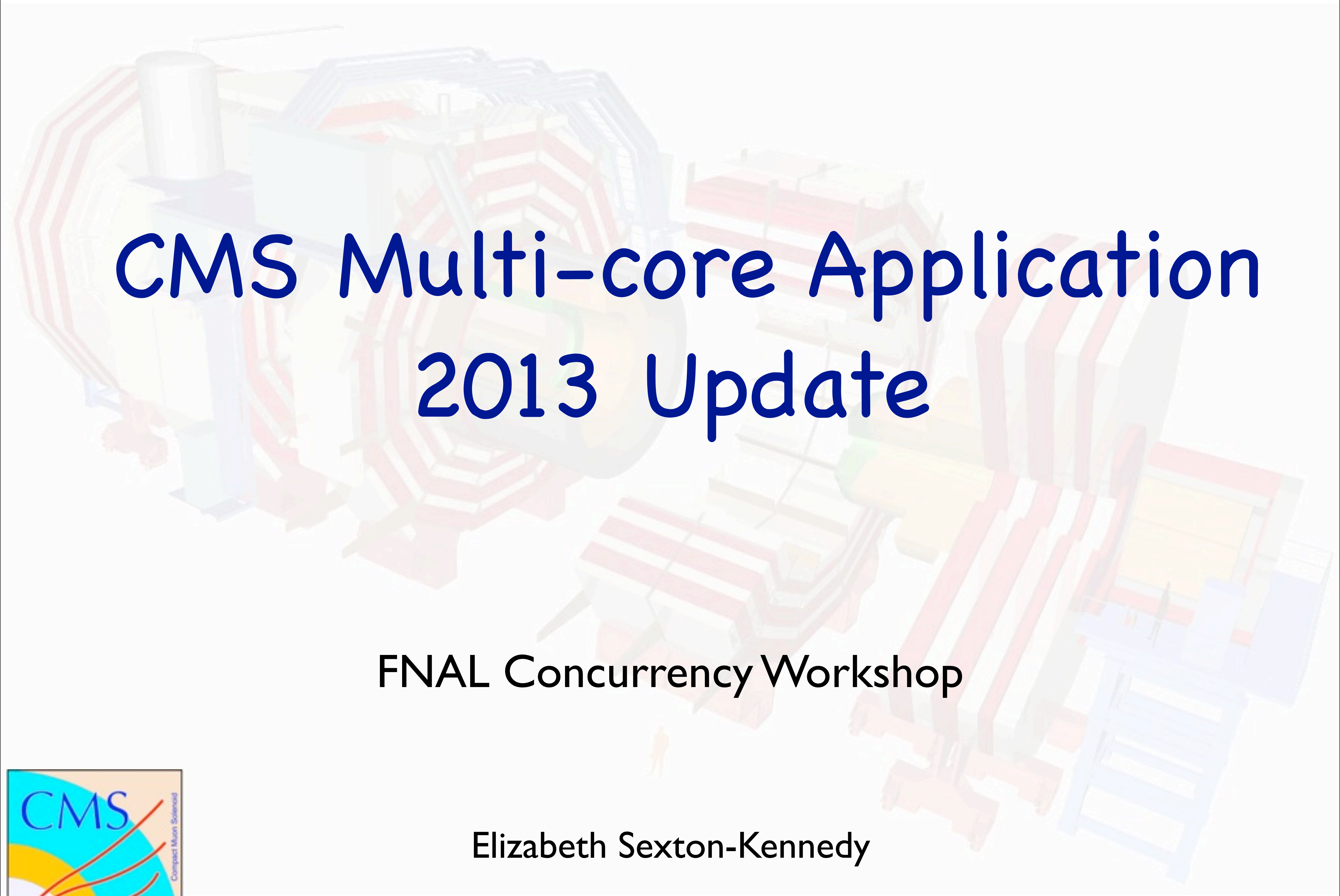


CMS Multi-core Application 2013 Update

FNAL Concurrency Workshop

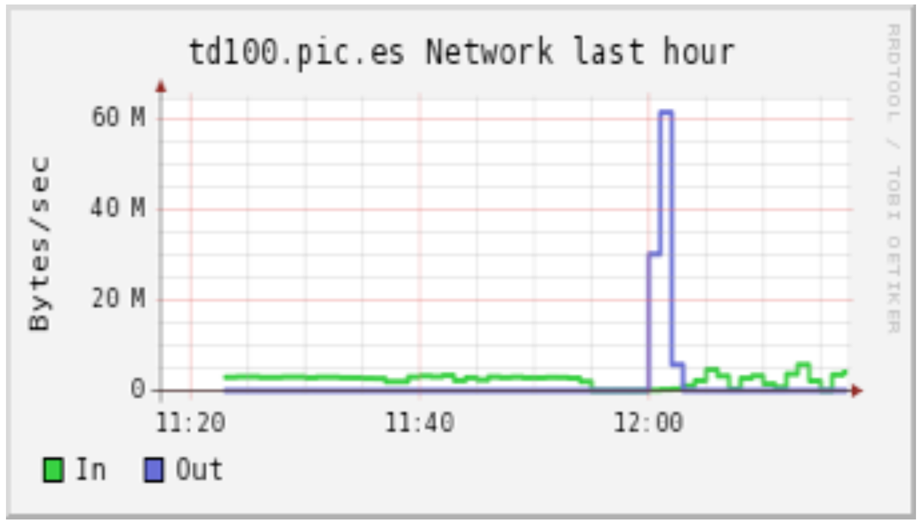
Elizabeth Sexton-Kennedy

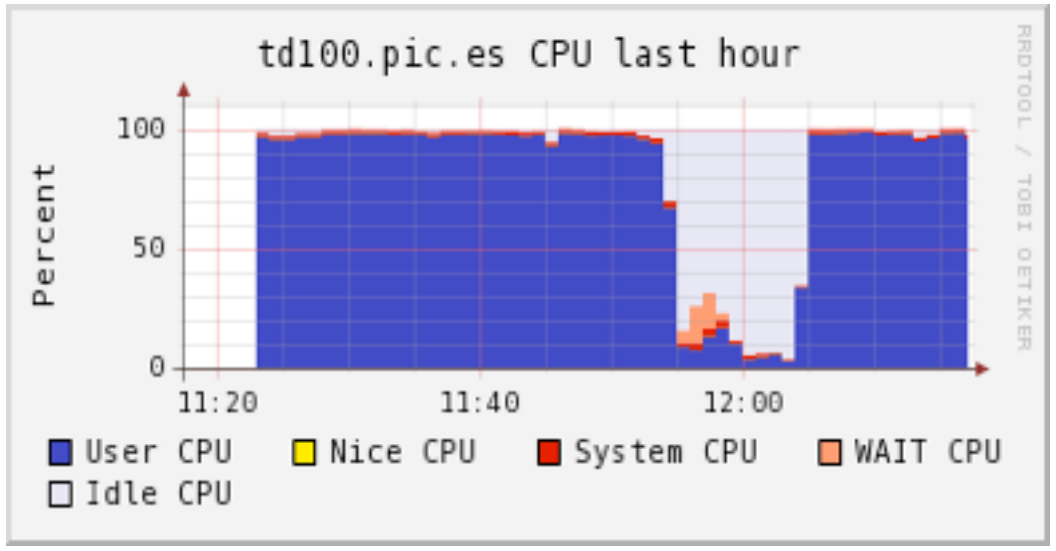
Introduction

- This is a current status and plans talk relative to the talk I gave in Nov. 2011. Much of what I said is still true, I'll point out the differences and changes.
- Topics include:
  - Review of Whole Node Deployment
  - CMS Code Performance in 2012
  - Requirements of a Fine Grained Parallel Framework
  - On Going Prototype Work, Milestones and Plans
  - Opportunities for Collaboration

Review of Whole Node

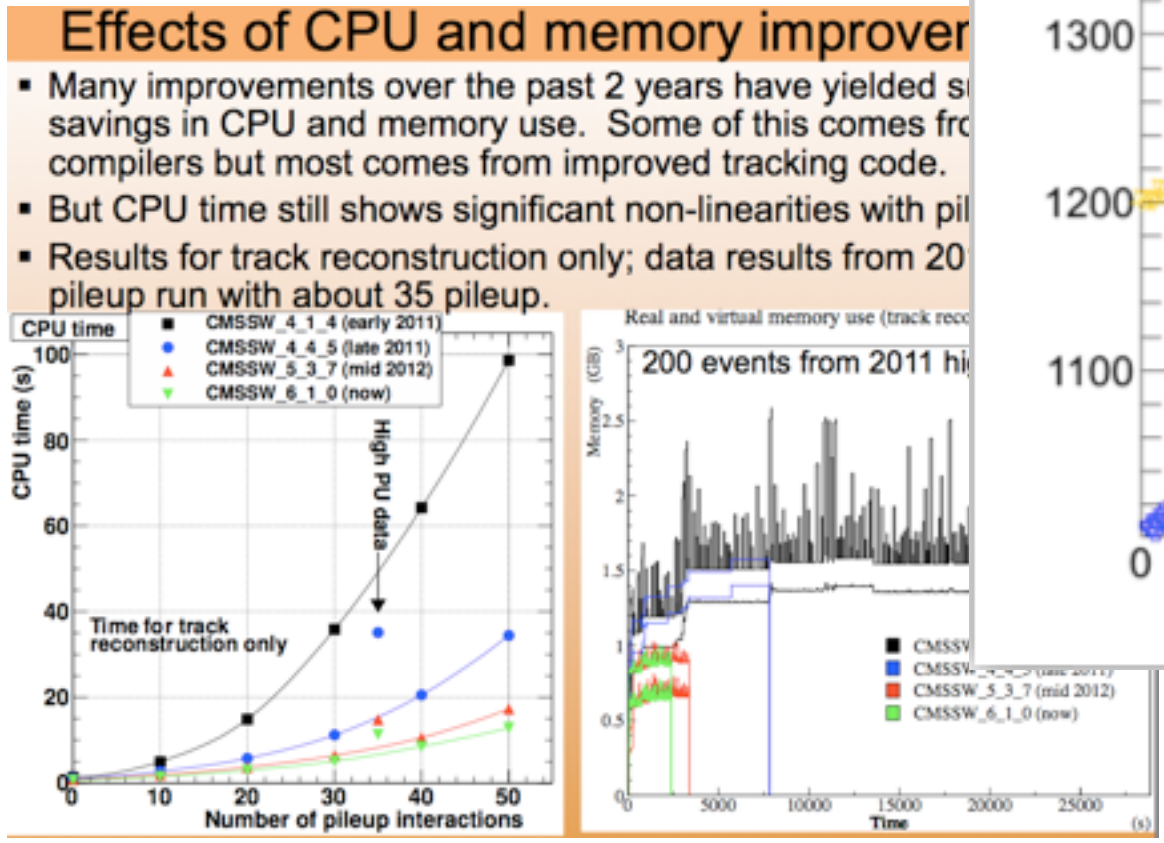
- Last year Computing Integration did GRID testing of our existing multi-core application based on Fork-Copy-On-Write
  - The tests were successful, the measured overhead due to children processing dispersion, file merging and stage-out (~10 minutes all cores ~idle) was deemed small (relative to ~8hr. jobs).
  - Compared with normal single-core jobs, the asynchronous merging and number of processing jobs needing to be tracked by WMDM were very much reduced.
  - Memory consumption for an 8 core job was reduced by 20%. Job memory performance including PSS was correctly reported.
- What went wrong? Most Tier-1 sites not willing to significantly increase dedicated resources in whole-node queues and CMS code changed requirements
  - Preparing GRID resources for whole node deployment is a target for 2015.





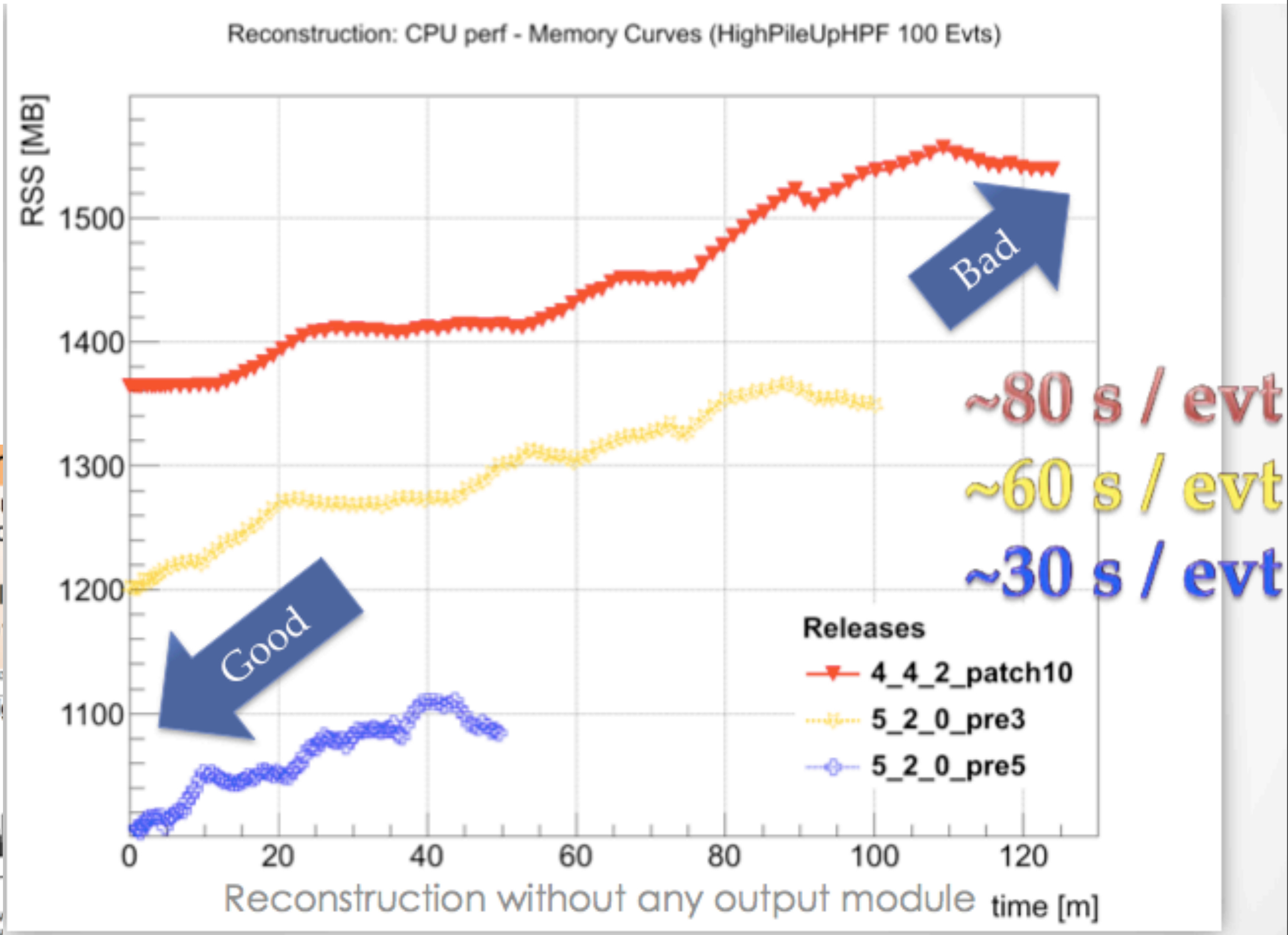
CMS Code Performance in 2012

- In Nov. of 2011 Offline's biggest concern was the predicted doubling of the luminosity and knowing that the release planned for data taking (5_1_0 incremental changes to 4_4_0) was not good enough. The need for 5_2_0 was clear.

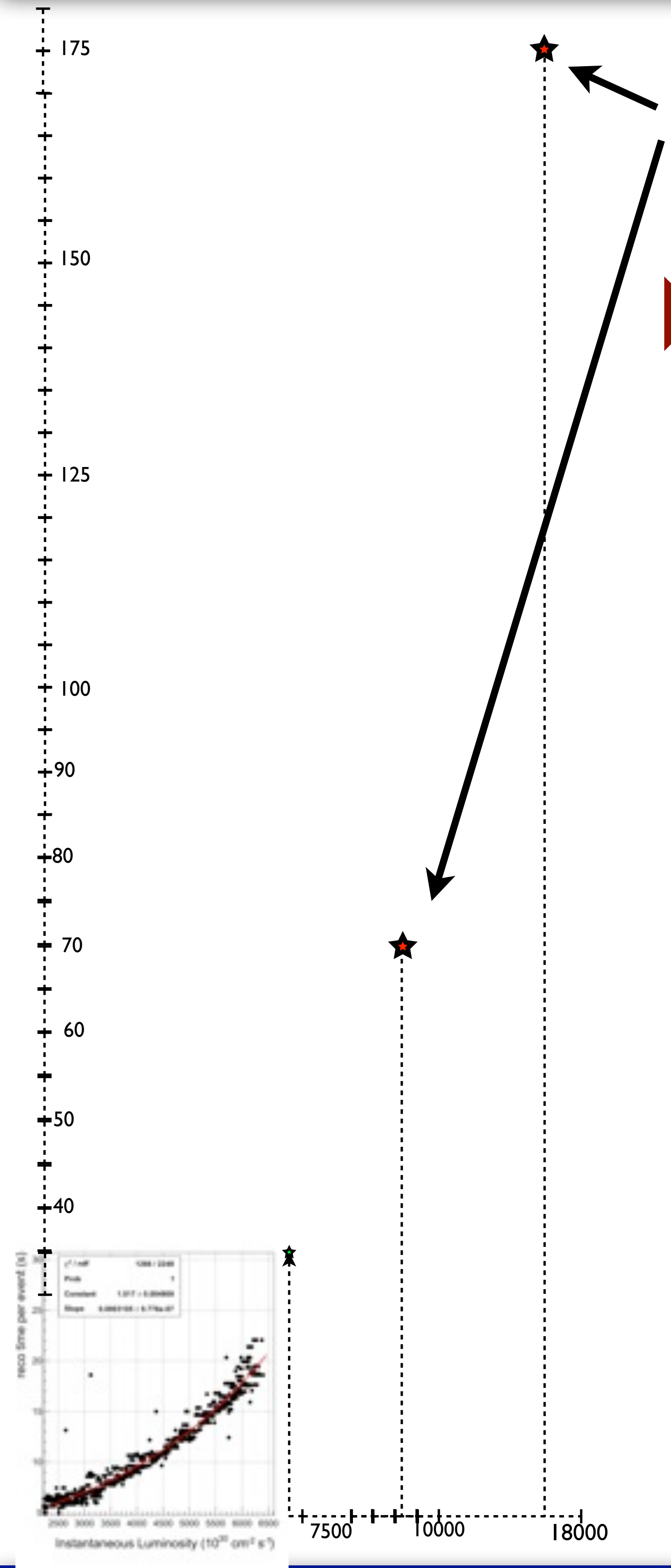


Effects of CPU and memory improver

- Many improvements over the past 2 years have yielded si savings in CPU and memory use. Some of this comes fro compilers but most comes from improved tracking code.
- But CPU time still shows significant non-linearities with pil
- Results for track reconstruction only; data results from 20 pileup run with about 35 pileup.



The Broader Context



Points measured from current release run on high PU MC

- There are 4 major tasks for CMS Offline in LS1:
  - Rewriting the core software to support fine grain parallelism
  - New tracking - New algorithms, improvements to performance, and thread safety... Particle Flow is probably next on the list
  - Improve the performance of Geant4 by optimizing CMS's use of it
  - Support Upgrade developments for both phase1 and phase2, we are considering supporting 3 tools for this purpose

Requirements for File Grained App.

- With only 2 years we can not afford a big bang change to a new framework. We need an evolutionary approach not revolutionary one.
- The performance of the multi-threaded application can't be worse then running several independent processes simultaneously.
  - The concurrency work has to be done simultaneously with the technical performance work.
  - The physics performance must remain the same despite the PU challenge.
- The new framework must minimize the amount and cost of code migration for the non-framework code.

Milestones and Plans

- In the May release we will have the new user interfaces implemented in the single threaded framework and we will finalize the implementation design of the new framework which will be based on TBB.
- In addition the framework data structures in the single threaded framework will be made thread safe.
- We will centrally identify thread un-safe inter-module data products for event and conditions data.
  - We are building tools to analyze the full code base of CMSSW look for thread safety problems.
  - As part of the build a checker would run to keep it thread safe.
- In the Nov. release the thread based scheduler will be swapped in.

Opportunities for Collaboration

- Things we want from community and SFT supported software:
  - thread safe histogramming
  - thread safe root I/O
  - thread safe Geant 4 without a performance penalty

Summary

- Scaling current throughput up to future commodity computing platforms is our long term goal.
  - We no longer think forking will be sufficient, so we are migrating to a finegrained parallelization solution managed by framework.
  - I/O bottleneck? Memory bottleneck?
  - Focus is on scaling production jobs for future hardware
- View upcoming shutdown in 2013 as only opportunity to introduce significant changes in the framework.
- Unlike last year we now know that there needs to be significant changes to physics algorithmic code.