Using Palm For Analytical Performance Modeling

NATHAN TALLENT, KEVIN BARKER, DARREN KERBYSON, ADOLFY HOISIE

Pacific Northwest National Lab

Workshop on Modeling & Simulation of Systems & Applications

August 14, 2015

Tools for Analytical Modeling of Performance

N. Tallent & A. Hoisie. ICS 2014

annotated source → Palm Compiler → reference & instrumented executables → Palm Monitor → profiles → Palm Generator

Palm Compiler → static analysis → Palm Generator

Palm Generator → model (program) ← parameters

model (program) → prediction & diagnostics

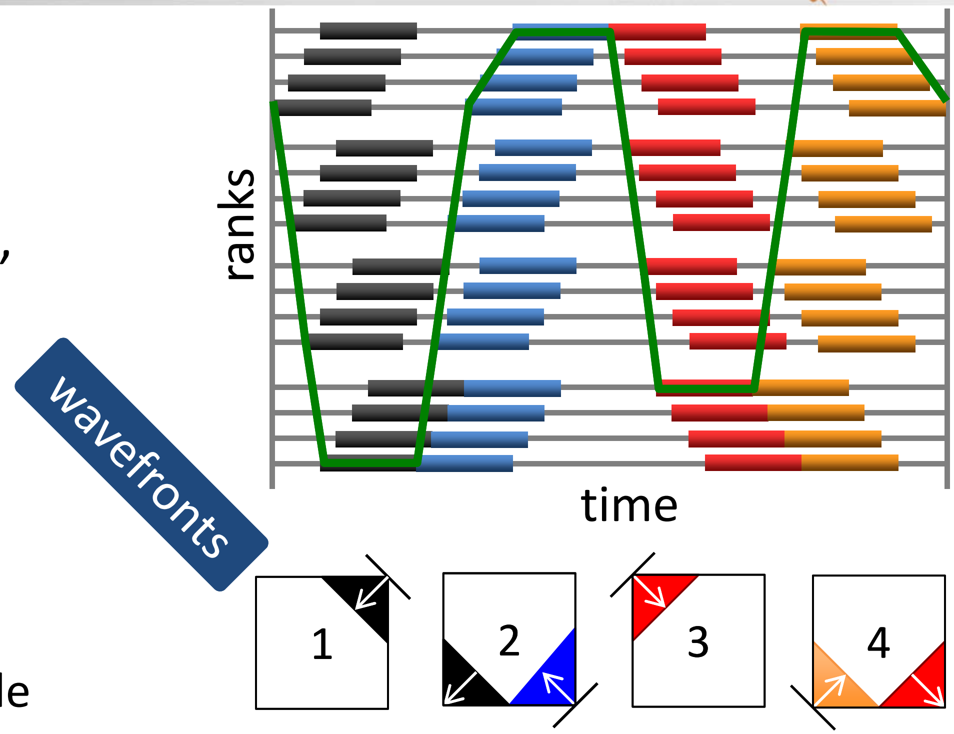
- Application modeling is difficult
  - Creating & validating models requires expertise and labor
  - Reproducing & distributing models is ad hoc
- Palm: PAL Modeling tool
  - Annotation language expresses insight & guides modeling
    - Develop model and application in tandem
    - Decompose modeling task into sub-problems
  - Generate model from static/dynamic annotation structure
    - Name, capture, and use dynamic application structure
  - Generate same model given same input (reproducible)
  - Generated model is executable program (distributable)

refine as necessary

Modeling a Wavefront Application: Sweep3D

- Need more than static analysis
- Sweep3D: 2D pipeline
  - Wavefronts propagate in phases, yielding active and idle states
  - Runtime depends on phase, pipeline shape, & pipeline stage

    f(shape, phase, stage)
  - Pipeline formed dynamically
    - state variables and guarded code



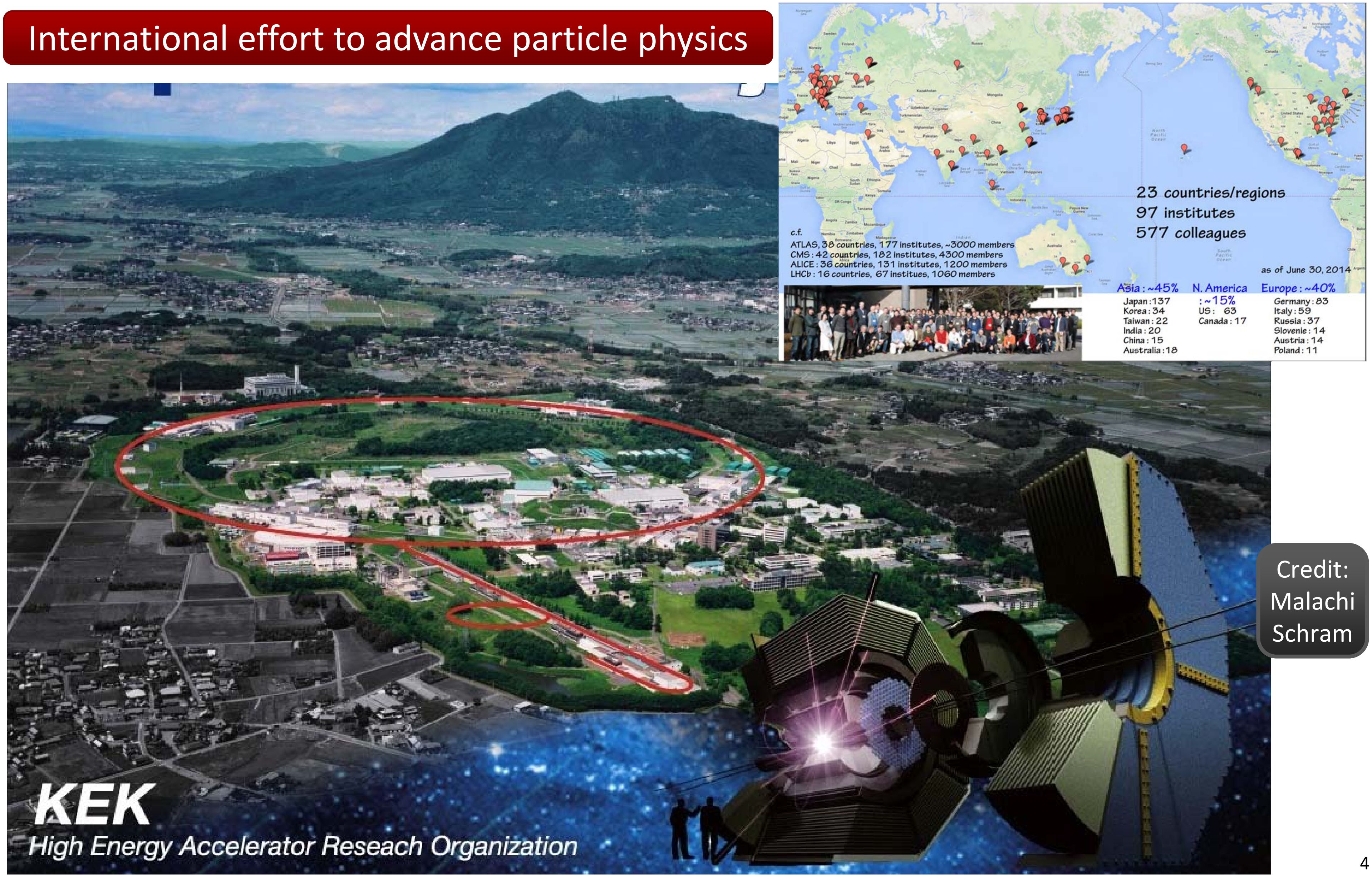
- Palm assists modeling the critical path – before it exists
  - express idle time as function of a pipeline stage's model
    - model critical path using a forward reference to a generated model
  - Palm assembles model using dynamic analysis & composition rules

f(shape, phase, *M*(stage)) → *f*(shape, phase)

human   tool

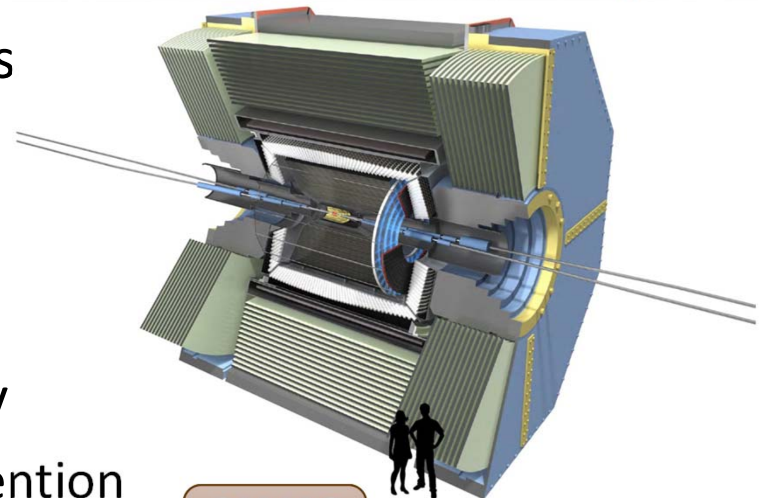
High Energy Physics: Belle II Analysis Workflow

International effort to advance particle physics



Credit: Malachi Schram

Predict & Mitigate Contention in Belle II Workflow



- Belle II Workflow: Extensive data analysis
  - Normalize data and 'do physics'
- Data! 25 PB/year of raw data
  - Stored data expected to reach 350 PB
- Many analysis pipelines run concurrently
  - Goal: Predict (& mitigate) resource contention
- Example analysis pipeline:

iTOP data → iTOP Converter
Pedestal file → Pedestal Tool
CAMAC data → CAMAC Converter → ChanTime Calibr. Tool

iTOP Converter → Pedestal Tool → WaveMerge Tool → WaveQaulity Tool → WaveFitter Tool → ChanTime Calibr. Tool → SampTime Calibr. Tool

SampTime Calibr. Tool → ROOT Tree Gen Tool
SampTime Calibr. Tool → Monitoring Tool

  - Dynamically assembled modules (Python script)

Palm creates workflow model by composing models for each module

Assessing the Impact of Silicon Photonics

- Question: What is the impact of silicon photonics on graph-based workloads in the 4–6 year timeframe?
- Methodology
  - Work with architects; Identify silicon-photonics enabled systems
    - IBM TOPS (64 nodes, fully connected): photonics off node
    - Oracle Macronode (32 nodes, fully connected): photonics on & off node
  - Select workloads bound by network bandwidth and latency
  - Compare silicon-photonics systems with electrical counterpart
    - fix footprint; fix power
  - Large, distributed graphs ("require a rack")
    - Validate at scale 34; Project at scale 40
    - Scale $\stackrel{\text{def}}{=} \log_2(\text{edges})$
  - Models explore both performance and power
  - Model intra-node and inter-node data movement

Two Workloads To Represent Important Use Cases

Community Detection

- Input: Graph with weighted edges
- Output: Disjoint sets of related vertices
- Aggregated personalized all-to-all to send each edge's target info (~1 GB)
- Iterate until Δ-modularity < threshold
  - Each vertex initially its own community
  - For each vertex, determine whether modularity increases by moving to neighboring community

Large, aggregated messages

- Optimized for cluster networks
- Combine reqs with same target vertex

More computation

- Modularity requires collectives
- Denser graph; aggregation cost

Using Palm…

Annotations convey insight about input graph

Capture important runtime properties. E.g.: probability that communities are formed

Swap network models

Convenient representation

Challenge: Help specialize model for graph input class

Conclusions

- Ease burden of modeling
  - Facilitate divide-and-conquer modeling strategy
  - Automatically incorporate dynamic structure
  - Generate contribution and error reports
- Enable first-class models
  - Coordinate models and source code
  - Functions unify annotations, generated models, and measurements
- Expressive: elegantly represent non-trivial critical paths
  - Annotations provide convenience within fully generic framework
- Reproducible: generate same model given same input
  - Generate model according to well-defined rules
  - Define model structure from static & dynamic code structure
- Future: Especially interested in more dynamic assistance