MPI and Scalable Parallel Performance Analysis

25 Years of MPI Workshop, ANL, September 25, 2017

DCG
Data Center Group

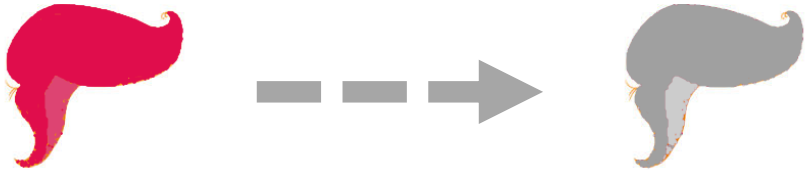
Fair Warning/Disclaimer

The views expressed by this presentation are mine and not necessarily those of Intel

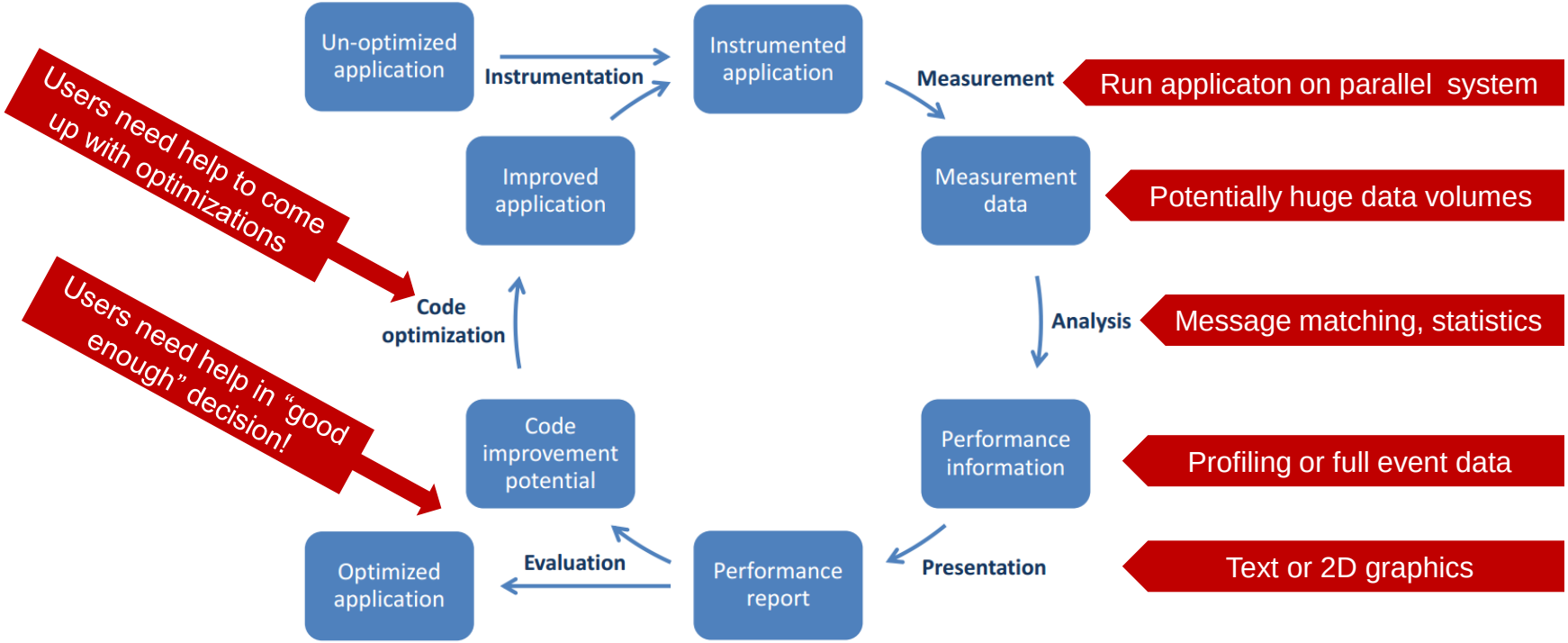
Lots of the graphical material shown comes from parallel tools projects (ANL/Jumpshot, BSC Extrae/Paraver/Dimemas,, JSC/Scalasca, Score-P, TU-D/Vampir, …)

I'm ignoring most of the more lightweight, pure profiling tools

Getting involved with MPI can lead to this:



The Performance Analysis/Optimization Cycle



Scalable Parallel Performance Analysis Today

Several well-established portable tools families

- Europe: BSC Extrae/Paraver, JSC Scalasca, TU-Dresden Vampir
- US: TAU

Very few commercial tools

- Know only of Intel Cluster Tools and Allinea/ARM MAP

Scalability to the 100000s of processes

- Both Scalasca and Vampir have demonstrated this
- Key is to use massive parallelism for analysis and (graphical) presentation

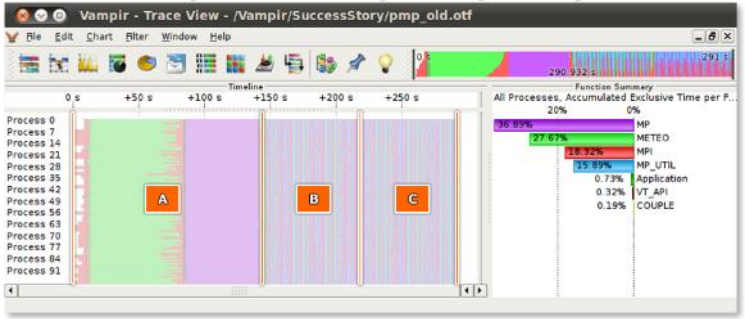
On- and off progress in tools integration/interoperability

- Score-P as latest effort covers TAU and two European tool families
- Generally, data file formats can be converted

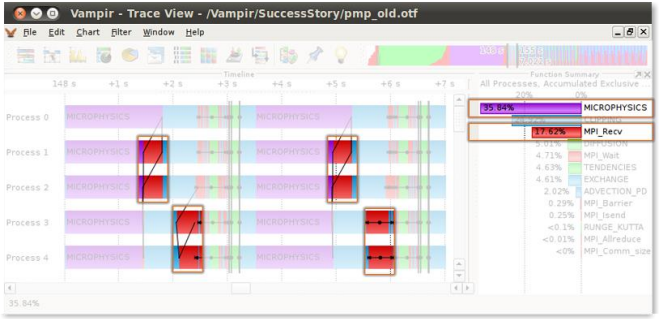
Limited progress in correctness checking & modeling

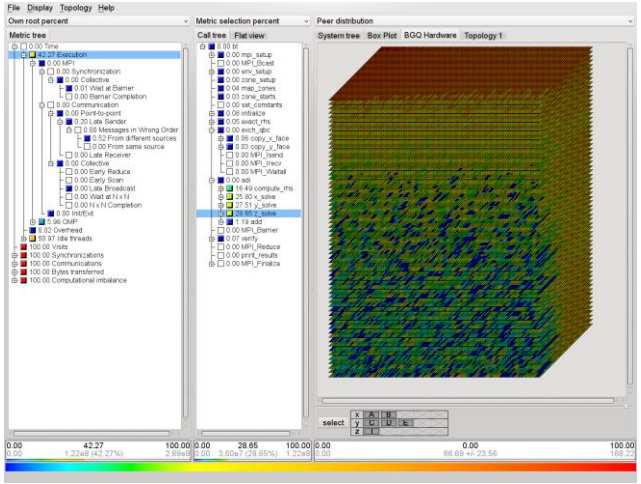
- Tools tend to report the status quo, can't extrapolate or answer "what if" questions
- MPI semantics are a harsh mistress – most mistakes keep your code from working (debugger time)

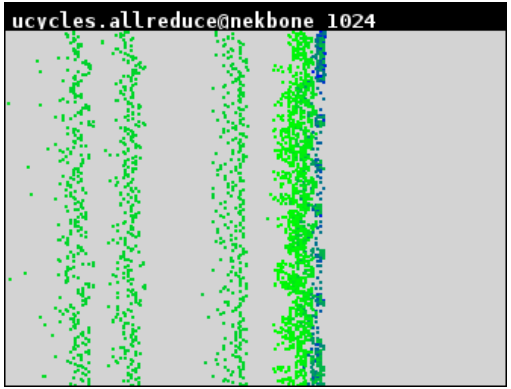
A few Colourful Graphics

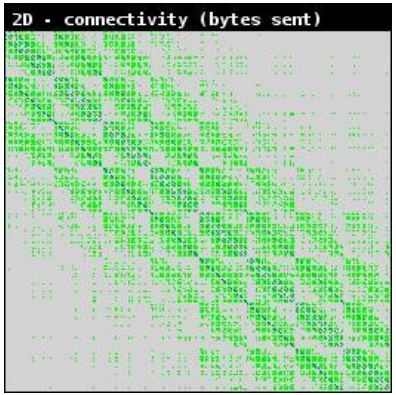


Zoom







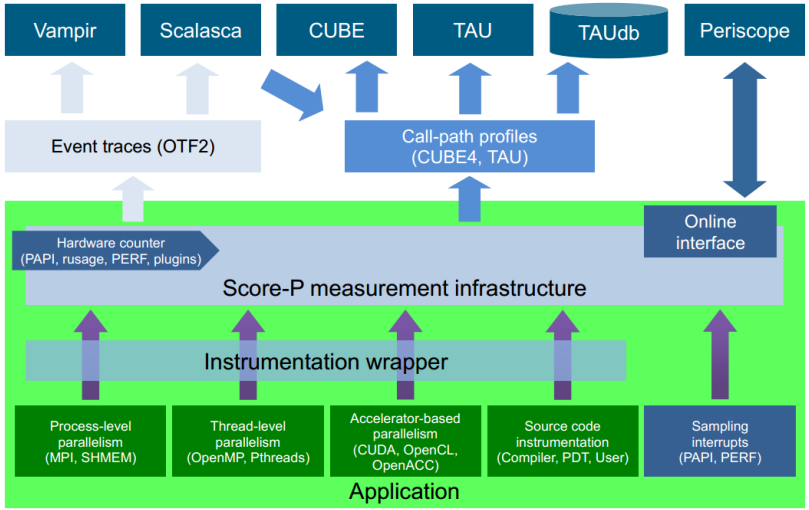


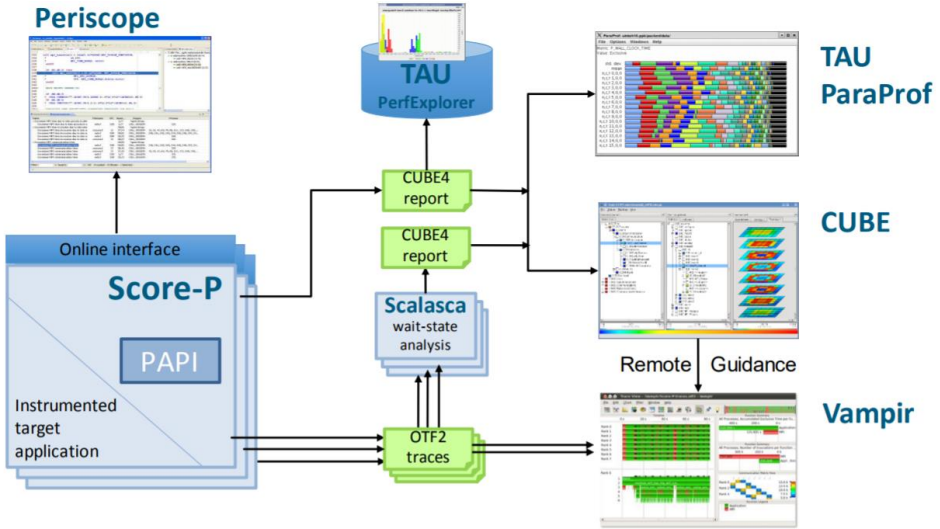
Integration and Interoperability

Data interoperability is key here - want to be able to look at the same trace data through different lenses/tools

- Not rocket science, yet trace formats have become quite complex

US/European Score-P initiative





MPI Advancing the Tools Field – the Good

Profiling interface

- Provides a transparent & reliable way to intercept calls & record data
- Gives access to all application-visible MPI-related data

Communicator concept

- Enables clear separation of application and tools communication
- Critical to achieve reliable tool operation
- Tools did immediately use MPI internally

MPI debugging I/F

- Could build tools that attach to running applications

Market effects

- Portability → tools (reasonable easily) portable to all systems supporting MPI
- Users → reach a vastly larger user community due to only one “message passing model” for all systems & applications
- Clear & orthogonal semantic → reduce effort required for analysis code

MPI Advancing the Tools Field – the (Slightly) Bad

Limited MPI introspection

- Can't see "inside" the MPI calls or the progress engine
- Some analysis questions are hard to answer
  - Why is MPI call XYZZY taking so long?
  - How much time is taken up by MPI SW stack vs. network stack & transmission?
- This is a bigger problem for MPI-2 "one-sided" operations, f.i.

No way to record message matching

- To match sends and receives, all tools replay the MPI message matching rules
- This can break down when watching only parts of application runs

Are we Done Yet?

Couple of hard problems do remain

- Trace data deluge
- End-user information overload & required expertise
- Answering the *real* end-user questions
  - How good is my code, and what could optimizations achieve?
  - How well does it scale?
  - How will it run on a different system?

Couple of ideas/references in the following slides

Data Deluge – On-Demand Trace Collection

Tracefiles are *always* too big (recent example: 3 TB for a NLP ML application on 64 processes)

- Want to be able to safely en/disable tracing without screwing up message matching
- Want to be able to safely cut recorded tracefiles
- Prefer automatic triggers to assist

Before you ask

- End-users often unable/unwilling to cut down workloads

Ideal world

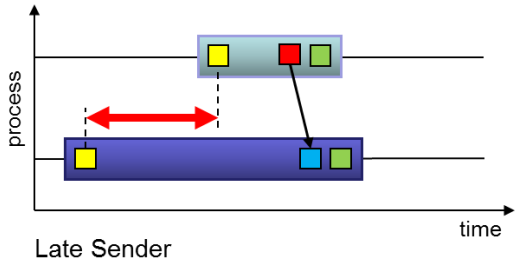
- Traces are collected when a performance metric indicates a problem
- Lightweight monitoring produces the underlying data, ML techniques as trigger

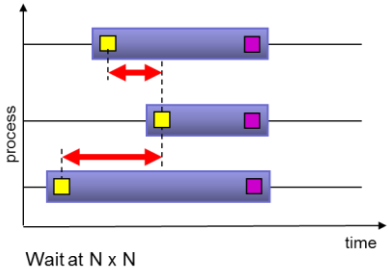
Identifying Performance Problems

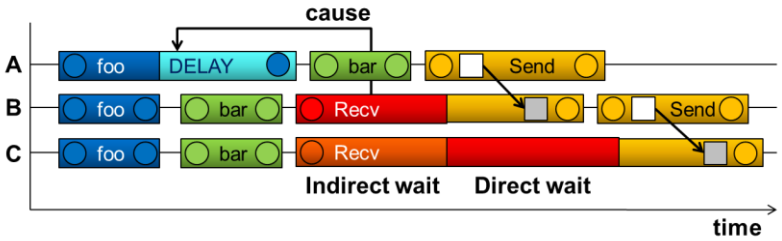
Current tools largely assume a “black belt” expert in the drivers seat

- This seriously limits their take-up

Some tools try to identify MPI bottlenecks and link to root causes in the program (Scalasca as example)







This pretty much requires replay of an application

We need more of this

- Hard-coding rules does not scale at all
- ML techniques could have a role here

Modeling & What-If Scenarios (1)

BSC Dimemas replay tool

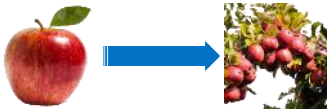
- Replay application run with CPU scaling and communication model
- Assess impact of MPI implementation and interconnect: replay with BW=∞, Latency=0 & no contention → this has proven to be very useful

BSC multiplicative performance model

- Partition parallel efficiency into three factors

$$\eta_{||} = T \times LB \times \mu_{LB}$$

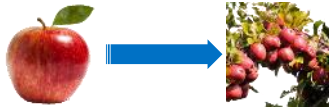
  - Transfer (T): effect of the interconnect network
  - Load balance (LB): difference in work between processes
  - Serialisation (μ_{LB}): process dependencies and transient load imbalances



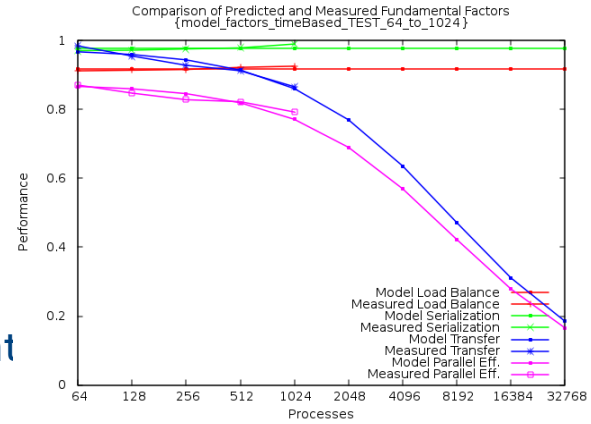
Division of responsibilities

- μ_{LB} and LB are the application developer's problem
- T can be addressed by MPI and system developers

Modeling & What-If Scenarios (2)



Fit &extrapolate efficiency factors, usually resulting in depressing predictions



Scalability analysis with Extra-P

- Measure and fit the scaling behaviour of code component (block, MPI calls)

$$R_n(x) = \sum_i a_i x^{\frac{b_i}{c_i}} + \sum_j \log_2^{d_j}(x)$$

- Scaling model is the sum of all component models

$$R(x) = \sum_n R_n(x)$$

- Integrated wit Scalasca infrastructure
- Give it a try!

Where MPI Could Help in the Future

Wish #1 – Message matching

- Avoid need to replay communication and re-match messages
- MPI-internal mechanism or ways to extend the message header
- Fundamental to address the data deluge

Wish #2 – Addtl. Introspection (MPI_T?)

- Collect data on separate (logical) phases in MPI operations
- Examples
  - Data type processing vs. transmission of serial byte stream
  - Completion of one-sided operations
  - Data volumes in and out for collectives
- Callback method preferred
- Prescribed, strict semantics??

The POP Project

POP – Performance Optimisation and Productivity

- European govt.-funded project (term 2015-2018)
- Partners include BSC and JSC as tools providers

Objectives

- Promote best practices in parallel programming
- Offer services to
  - Gain detailed understanding of application and system behavior
  - Propose how to refactor applications in the most productive way
- Cover academic as well as industrial users
- Support MPI and/or OpenMP

Success so far

- 72 performance audits, 5 completed PoCs (36 and 8 are WIP)
- Very favourable feedback from customers …

Semi-Useful Links …

Argonne MPI performance tools
- https://www.mcs.anl.gov/research/projects/perfvis

BSC performance tool suite
- https://tools.bsc.es/

Vampir tool
- https://www.vampir.eu/

Scalasca tool
- https://www.scalasca.org/

Extra-P tool
- https://www.scalasca.org/software/extra-p/

Score-P effort
- https://www.vi-hps.org/projects/score-p/

POP project
- https://pop-coe.eu/