U.S. DEPARTMENT OF ENERGY

Office of Science

Quantifying Overhead in Today's Execution Models

John Shalf, David Donofrio, and Nick Wright
Lawrence Berkeley Natl. Laboratory
Berkeley, California,

Bob Lucas, Pedro Diniz Jacqueline Chame, Gene Wagenbreth
USC / Information Sciences Institute
Marina del Rey, California 90292

Introduction

- **Project Goal:** *Quantify the deficiencies of today's execution models*
  - *It is NOT our goal to propose or evaluate a NEW execution model*

- **Project Strategy:** *Carefully examine select subset of today's applications to*
  - Compare measurements of these algorithms realized using today's CSP execution model on today's hardware
  - Contrast with models for how fast these algorithms should ideally perform in absence of overheads and inefficiencies of CSP model.

- **Starting Problems (first year targets)**
  - Fusion Code / PIC codes
    - GTC: Already have compact application version. Compare async exec model for particle deposition to bulk-synchronous approaches. Leverages a lot of existing experience
  - Combustion CoDesign Center codes
    - **S3D:** PDE solver on block structured grid with explicit scheme
    - **Multigrid:** Standalone multigrid solver created by Bell team to foster collaboration with MIC and NVIDIA code teams. Could be reused to study message driven vs. SPMD models and auto-tuning for this important kernel
    - **LMC:** Full Adaptive Mesh Refinement code. PDE solver for each patch looks much like S3D. Focus on opportunities to exploit asynchronous execution models for an adaptive algorithm.

U.S. DEPARTMENT OF ENERGY

Office of Science

Hardware Trends

- **Hardware Trends are breaking abstractions we have come to depend on**
  - **Parallelism**: Assume modest growth in parallelism→ *But parallelism is now growing exponentially*
  - **Locality**: Assume flat/uniform communication costs → *But costs are increasingly hierarchical*
  - **Computational Complexity**: Assumes FLOP is metric to conserve → *But cost of data movement exceeding cost of FLOP*
  - **Byte/FLOP ratios:** Assumes same ratios for memory capacity and bandwidth will remain→ *But cost per bit of DRAM and Bit/second of bandwidth is increasing relative to cost of computation*
  - **Heterogeneity**: Assumes uniform execution rates across system → *But source of execution rate heterogeneity (noise) are increasing drastically*
  - **Reliability**: Assumes reliable hardware *(or reliable enough)* → *But transient error rates increasing*
  - **Regularity**: Assumes non-adaptive/regular algorithms→ *But adaptive/irregular algorithms are the biggest growth opportunity for improved computational efficiency and new problems*

- **Result is reflected in growing performance gap (theoretical vs. delivered) and reduced performance portability**
  - Performance has been eroding for a long time now
  - Masked by increased effort in code tuning, but this path is unsustainable
  - Also masked by hardware/software ecosystem that has evolved to support the incumbent programming paradigm

Workplan

- **Key questions that need to be answered**
  - What underlying machine features are programming models currently ignoring
  - What is the power & performance consequence of ignoring those underlying characteristics
  - Or conversely, what is performance/power-efficiency opportunity for new execution model

- **Approach: Quantify sources of performance loss in current execution models**
  - examine a carefully chosen set of today's applications that together span some of the scientific domains and computational motifs that will be important for exascale
  - Develop models of how these applications would ideally execute
  - Compare and contrast these with measurements of the execution of these same algorithms when realized in today's CSP execution model, on today's hardware.

The result of this analysis will be a rigorous, quantified understanding of the overheads and inefficiencies introduced by the CSP execution model.

Metrics: Quantify Losses in Following Areas

- **Loss due to over-synchronization *(baseline is idealized computational balance)***
  - un-necessary synchronization (overuse of collectives because weak split-phase sync in MPI)
  - load-imbalances (different code phases have different work requirements)
  - serial work by MPI or OpenMP fork-join semantics
- **Loss due to ignoring data locality**
  - Poor management of vertical locality by automatic resources *(baseline is idealized data movement if it were explicitly managed)*
  - Poor management of horizontal locality such as loss due to ignoring distances between MPI tasks or topology of interconnect because it is not expressed in today's programming models *(baseline is optimal embedding of communication topology to machine hierarchy)*
- **Losses due to instruction address calculation and other non-FP operations or inability to schedule SIMD FP (what is the instruction mix?)**
- **Distributed memory overheads (how much redundant memory consumption due to need to replicate shared variables across distributed memory address space**

U.S. DEPARTMENT OF ENERGY

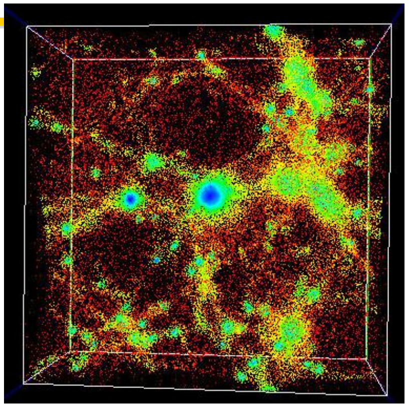
Office of Science

Technical Approach
Tool Capabilities

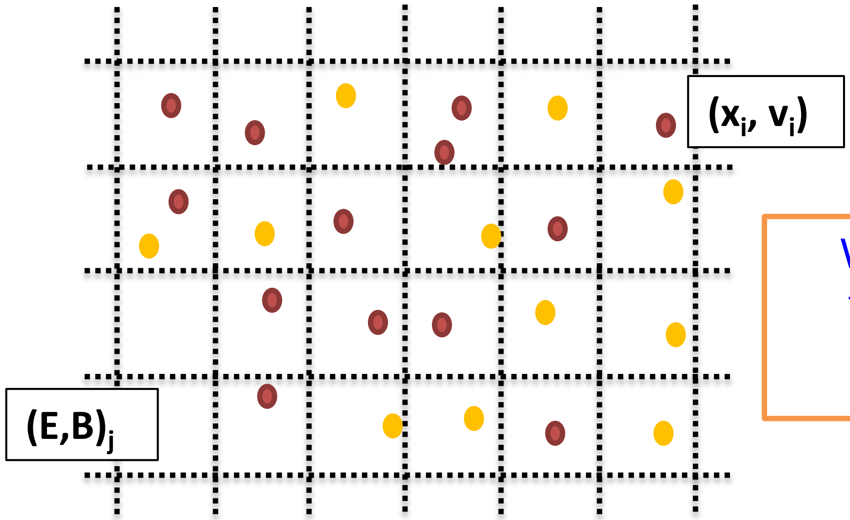
- **Algorithm Mapping Analysis**
  - Understanding high-level computations
  - Requires talking to the authors of the code
  - Characterize mappings to CSP models
  - Question: What are missing (unreachable) opportunities?
- **Execution Analysis (on existing machines)**
  - High-level concurrent and sequential partitioning
  - Coarse-grained concurrency analysis
  - Fine-grain concurrency analysis
  - Input/Output analysis
- **Leverage familiar tools on today's systems**
  - TAU & HPCToolkit
  - Research projects too (SLOPE)

U.S. DEPARTMENT OF ENERGY
Office of Science

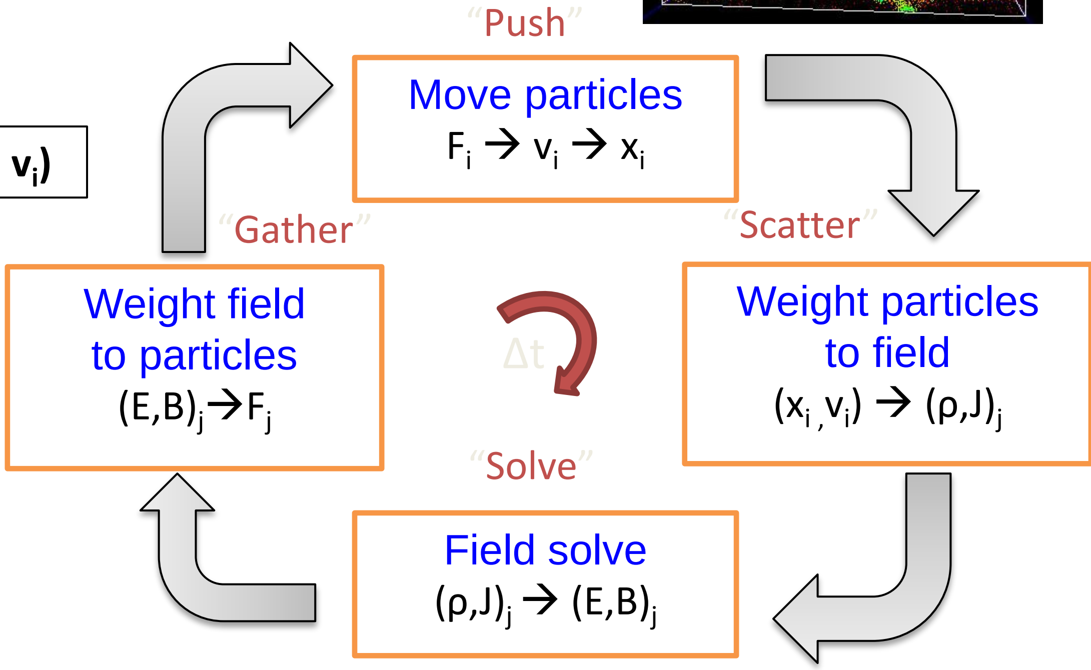
Particle-In-Cell (PIC): GTC



- Popular method for numerical simulation of many-body systems.
- Often implemented from first principles without the need of an approximate equation of state
- Applications: plasma modeling, Astrophysics and modeling of debris fields from explosions



Grid/mesh overlaying particles to measure charge and current densities



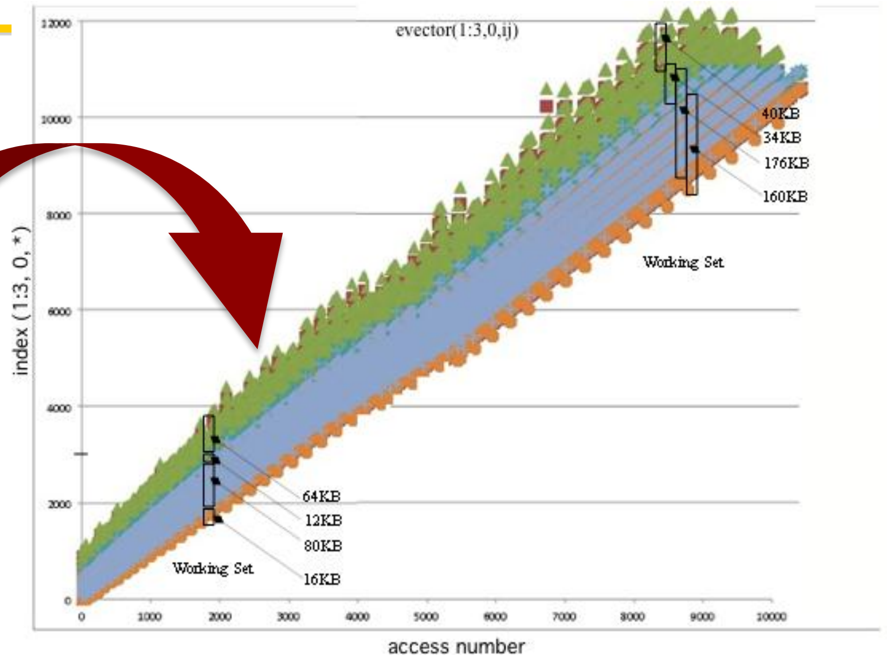
"Push"
Move particles
$F_i \rightarrow v_i \rightarrow x_i$
"Scatter"
Weight particles to field
$(x_i, v_i) \rightarrow (\rho,J)_j$
"Solve"
Field solve
$(\rho,J)_j \rightarrow (E,B)_j$
"Gather"
Weight field to particles
$(E,B)_j \rightarrow F_j$
Δt

Generic PIC Schematic

GTC: *pushe/pushi* Routine

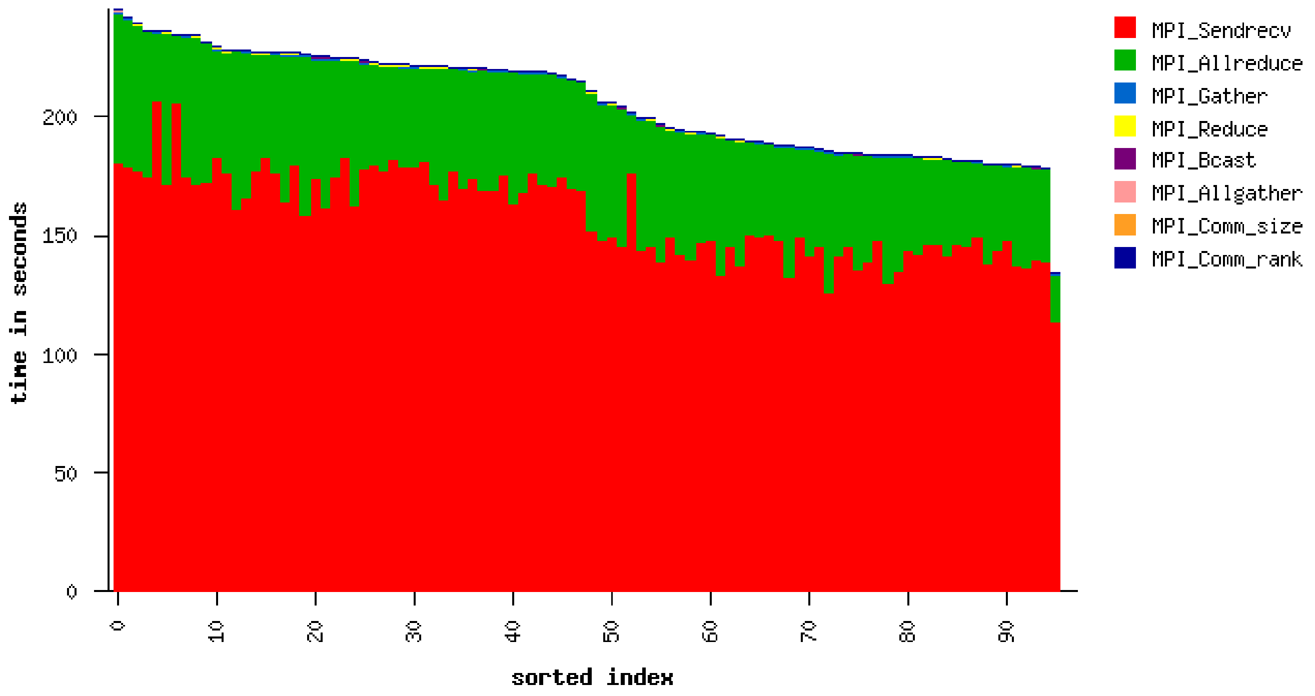
Source F90

```
do m=1,mi
  e1=0.0
  e2=0.0
  e3=0.0
  kk=kzion(m)
  wz1=wzion(m)
  wz0=1.0-wz1

  do larmor=1,4

    ij=jtion0(larmor,m)
    wp0=1.0-wpion(larmor,m)
    wt00=1.0-wtion0(larmor,m)
    e1=e1+wp0*wt00*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij))
    e2=e2+wp0*wt00*(wz0*evector(2,kk,ij)+wz1*evector(2,kk+1,ij))
    e3=e3+wp0*wt00*(wz0*evector(3,kk,ij)+wz1*evector(3,kk+1,ij))

    ij=ij+1
    wt10=1.0-wt00
    e1=e1+wp0*wt10*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij))
    e2=e2+wp0*wt10*(wz0*evector(2,kk,ij)+wz1*evector(2,kk+1,ij))
    e3=e3+wp0*wt10*(wz0*evector(3,kk,ij)+wz1*evector(3,kk+1,ij))

    ij=jtion1(larmor,m)
    wp1=1.0-wp0
    wt01=1.0-wtion1(larmor,m)
    e1=e1+wp1*wt01*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij))
    e2=e2+wp1*wt01*(wz0*evector(2,kk,ij)+wz1*evector(2,kk+1,ij))
    e3=e3+wp1*wt01*(wz0*evector(3,kk,ij)+wz1*evector(3,kk+1,ij))

    ij=ij+1
    wt11=1.0-wt01
    e1=e1+wp1*wt11*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij))
    e2=e2+wp1*wt11*(wz0*evector(2,kk,ij)+wz1*evector(2,kk+1,ij))
    e3=e3+wp1*wt11*(wz0*evector(3,kk,ij)+wz1*evector(3,kk+1,ij))

  enddo

  wpi(1,m)=0.25*e1
  wpi(2,m)=0.25*e2
  wpi(3,m)=0.25*e3

enddo
```



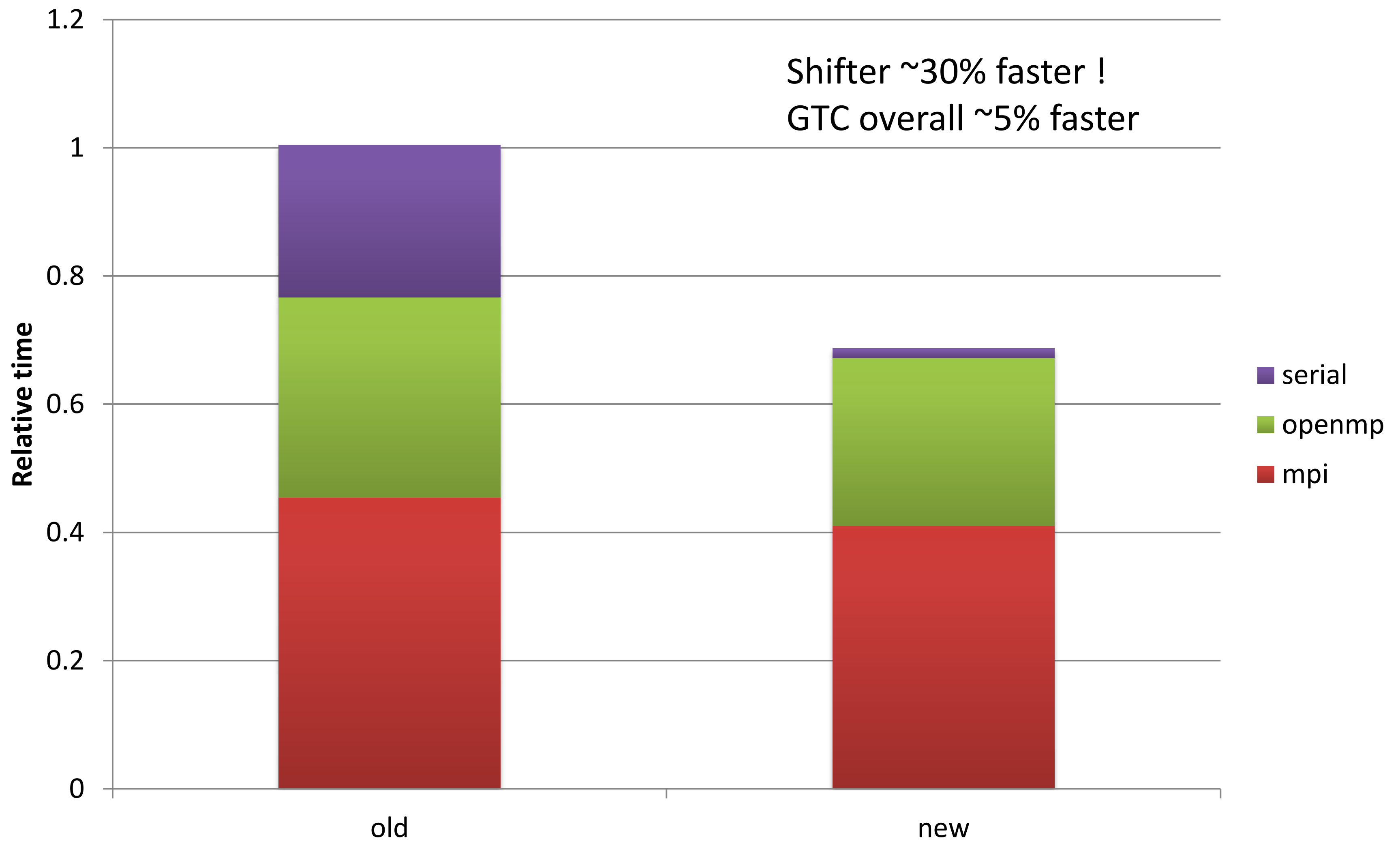
- Array Accesses:
  - Basic Stride-1 “Vector-Mode” Accesses
  - Many Indirect 3D Accesses by *evector*
    - lower dimension fixed (1,2,3,4)
    - middle dimension fixed with kk=0
    - upper dimension indirect with vector accesses

GTC: Load Imbalance



- Typical load imbalance performance loss can be between 10-30% of the runtime, depending on concurrency and problem definition
- Partially due to:
  - Dynamic nature of the computation- different numbers of particles move at each time step
  - Initial conditions – static load-imbalance

Tasking – (e.g. crazy things we do to overcome MPI serialization)



U.S. DEPARTMENT OF ENERGY

Office of Science

Thanks!

Come see our poster!