UNIVERSITY OF CALIFORNIA, RIVERSIDE

Optimizing Energy Efficiency for Distributed Dense Matrix Factorizations via Utilizing Algorithmic Characteristics

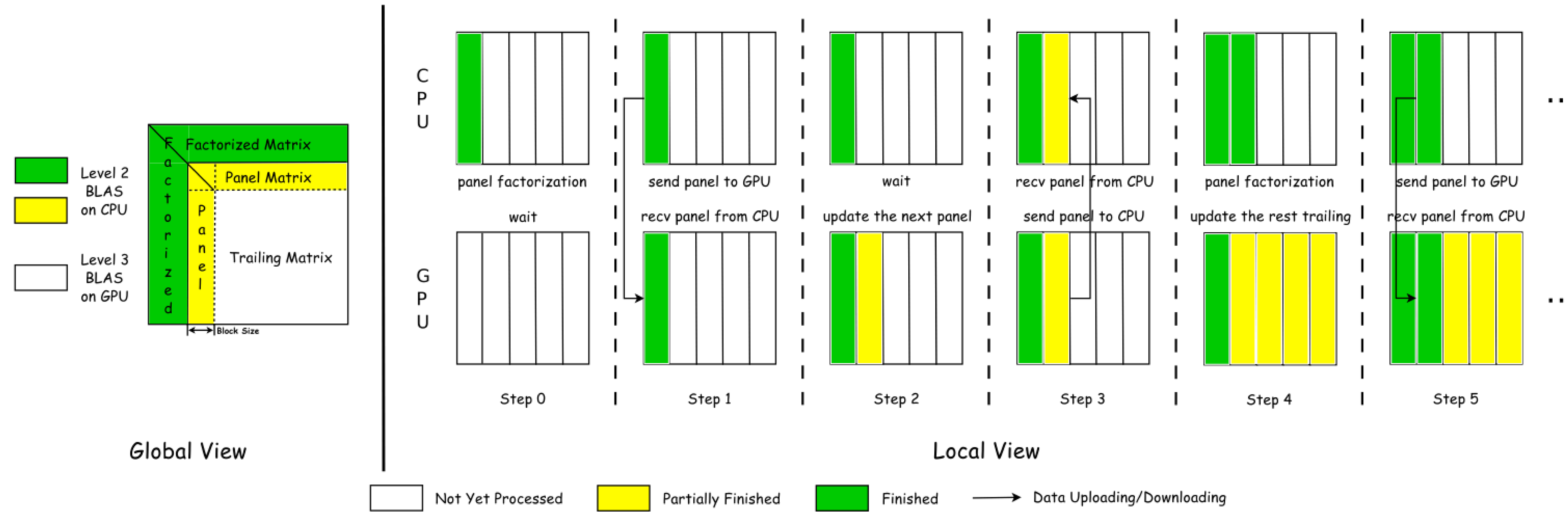
Li Tan and Zizhong Chen, UC Riverside

UNIVERSITY OF CALIFORNIA UCRIVERSIDE

Summary

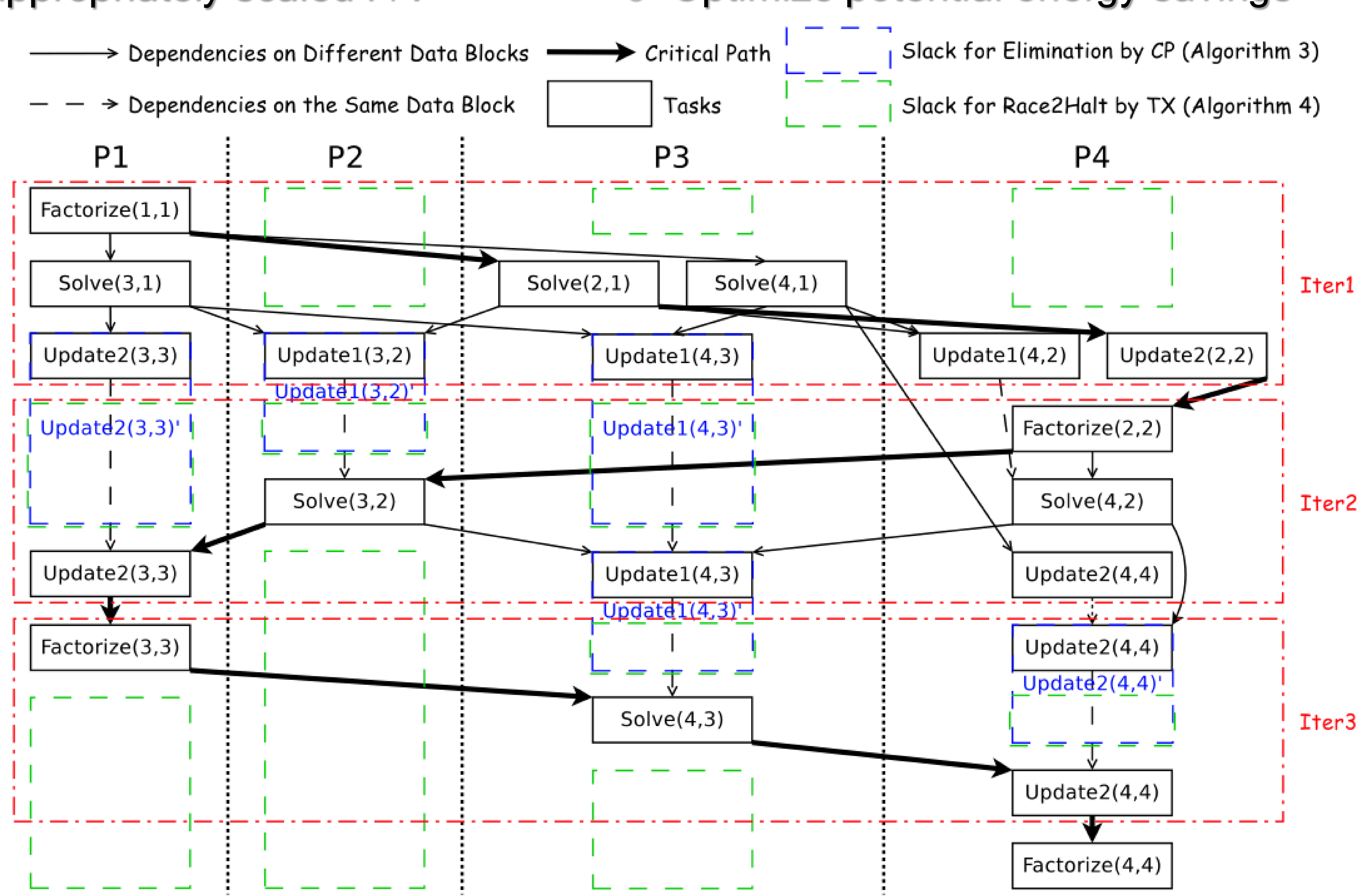
Goal: Achieve the optimal energy efficiency for linear algebra operations by using algorithmic characteristics

- Pressing demand of improving energy efficiency for high performance scientific computing
  - Scientific apps widely applied on supercomputers
  - Costs of powering supercomputers are increasing
- Strategic DVFS energy efficient scheduling
  - Switch ↓ hardware power if peak perf. not needed
  - CPU, GPU, memory → Handy APIs for DVFS
- Numerical linear algebra algorithms: LU, QR, Chol.
  - Dense Linear Algebra vs. Sparse Linear Algebra
  - Homogeneous Systems vs. Heterogeneous Sys.



Background: Energy Saving Strategies

- Critical Path (CP) and Slack Analysis
  - Slack: a period when a hardware waits for another
  - Slack Examples: load imbalance, network latency, communication delay, memory and disk access stall
  - Critical Path: a particular sequence of tasks where the total slack amounts to zero in task-parallel apps.
- CP-free Race-to-halt
  - Enforce hardware to run at the highest F/V when workloads are ready and at the lowest F/V otherwise
- CP-aware Slack Reclamation
  - Tasks on the CP: Run at highest F/V (peak perf.)
  - Tasks off the CP: Run at appropriately scaled F/V
- Detailed Comparison
  - Race-to-halt
    - Easy to implement
    - Hardware independent
    - Save additional energy due to load imbalance
  - CP-aware
    - Require CP detection
    - Hardware dependent
    - Generally the optimal
  - Hardware Utilizat. Polling
    - OS level monitoring
    - Coarse-grained
    - Runtime overhead



UCR

Overall Comparison: Existing Approaches

Operating Layer

- OS Level Approaches
  - Work aside app. at runtime
  - Make online decisions
  - No need of app-specific knowledge
  - General, no source modification
- Library Level Approaches
  - Customize energy saving decisions
  - Utilize app-specific knowledge
  - Library source modif. & recompilation
  - Optimize potential energy savings

Prediction Mechanism

- Online Slack Prediction
  - Online training of prediction model using execution info. in the same execution
  - Dynamic training overhead
  - Unexploited energy savings during training and prediction
  - Accurate prediction requires sufficient training: overhead ↑
- Offline Slack Prediction
  - Profile logged execut. history statically: no runtime overhead
  - Impractical for running large-scale app. in HPC: too costly

Architecture

- Homogeneous Systems
  - Slack arises among different processes through loop iterations
- Heterogeneous Systems
  - Slack arises between CPU and accelerators (e.g., GPU)
  - Similar techniques to HomoSys
  - Sleep state scheduling for unused/idle CPU cores

Algorithmic Energy Saving Strategy

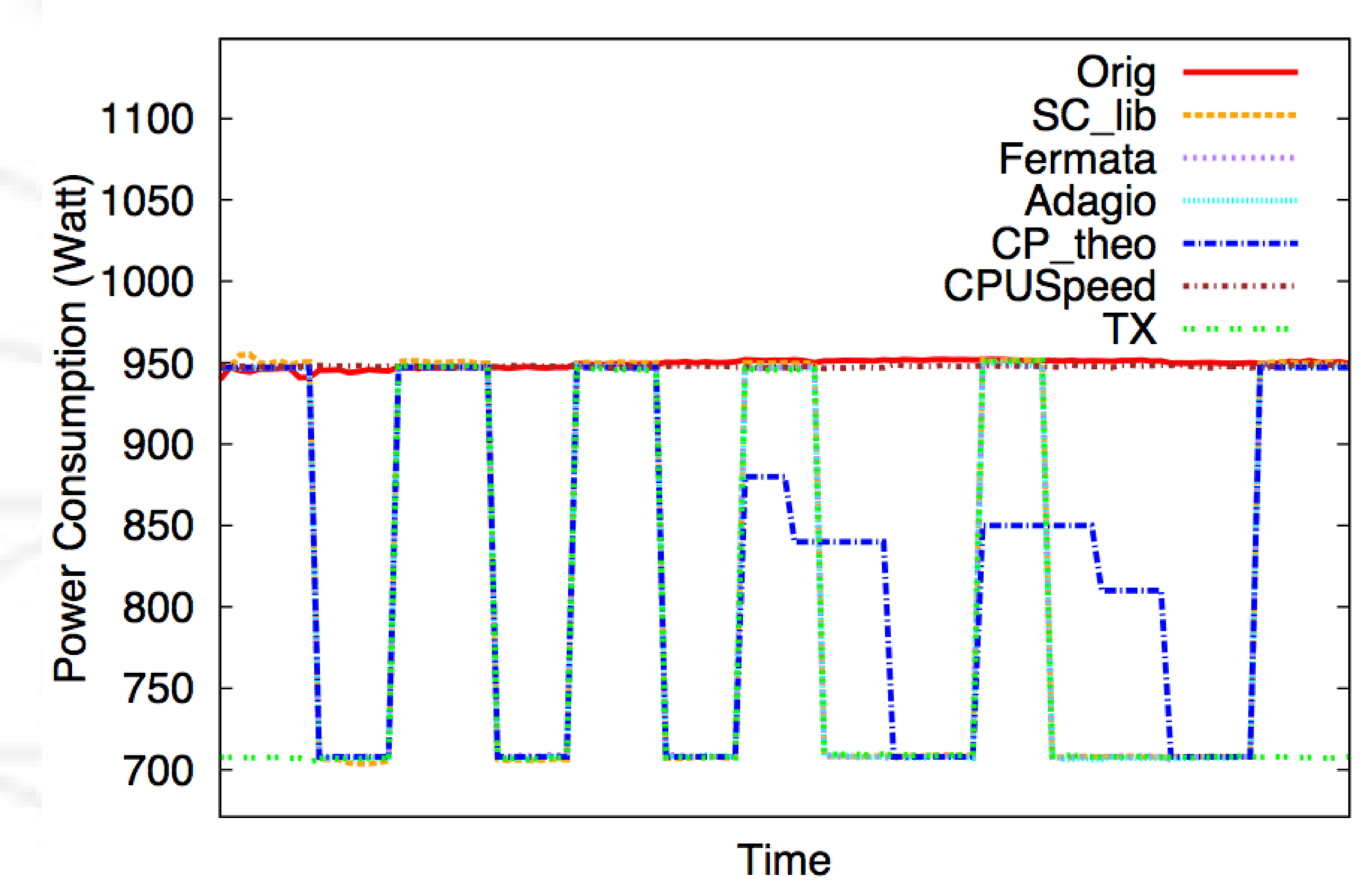
- Utilize Algorithmic Characteristics of Target Applications
  - Dense Cholesky, LU, and QR matrix factorizations
  - Algorithmic Task Dependency Set (TDS) analysis
  - Algorithmic online slack prediction

$$\frac{T^U_{i+1}}{T^U_i} = \frac{O(N'^3_{i+1})}{O(N'^3_i)} = \frac{(N-(i+2)\times N_{proc}\times nb)^3}{(N-(i+1)\times N_{proc}\times nb)^3}$$

$$slack_{i+1} = \mathbf{func}(T^U_i, T^U_{i+1}, param\_list)$$

Preliminary Results: Power Savings

Matrix Size: 160000 x 160000, Power Costs of Three Nodes



- Evaluated Power Saving Capability of Six Approaches
  - Distributed Chol. factorization with 16 x 16 process grid
  - OS level solutions failed to achieve the optimal savings
  - Race-to-halt can be comparable to CP-aware solutions