Using Intra-Core Loop-Task Accelerators to Improve the Productivity and Performance of Task-Based Parallel Programs

Ji Kim, Shunning Jiang, Christopher Torng, Moyang Wang, Shreesha Srinath, Berkin Ilbeyi, Khalid Al-Hawaj, and Christopher Batten

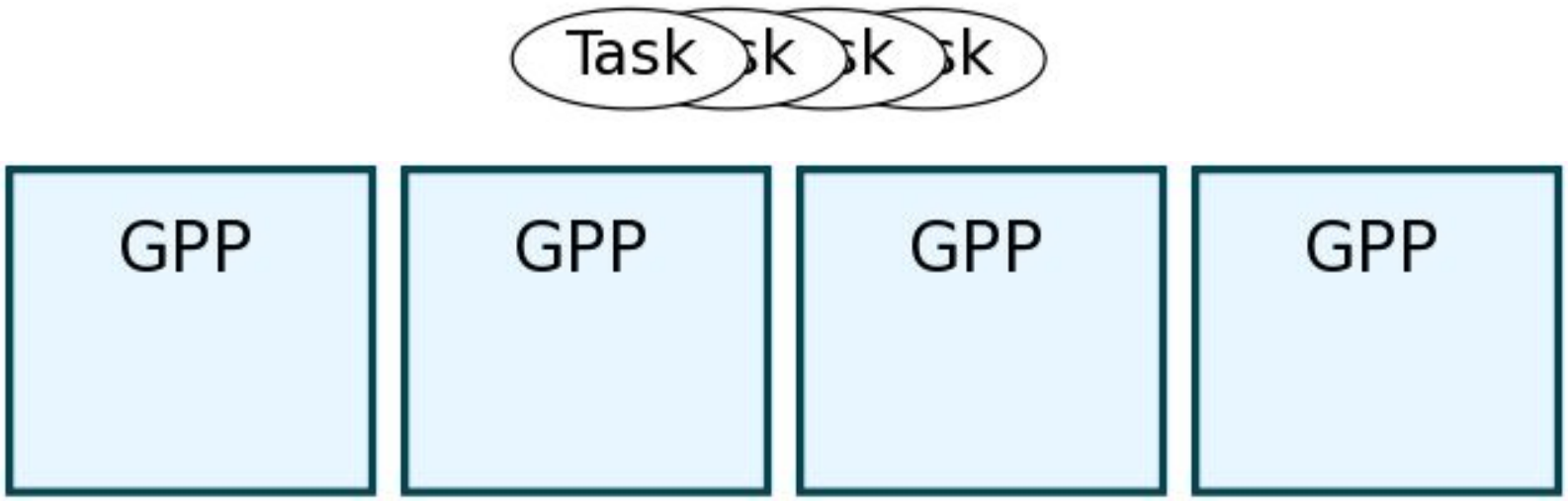
Computer Systems Laboratory

Cornell University

50th ACM/IEEE Int'l Symp. on Microarchitecture, MICRO-2017

Inter-Core

- Task-Based Parallel Programming Frameworks
  - Intel TBB, Cilk

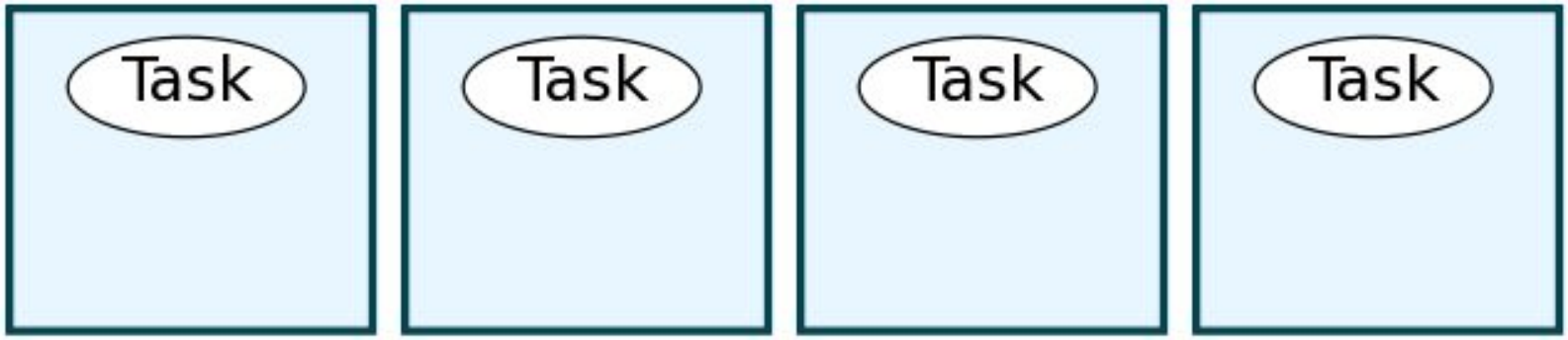


Intra-Core

- Packed-SIMD Vectorization
  - Intel AVX, Arm NEON

Inter-Core

- Task-Based Parallel Programming Frameworks
  - Intel TBB, Cilk

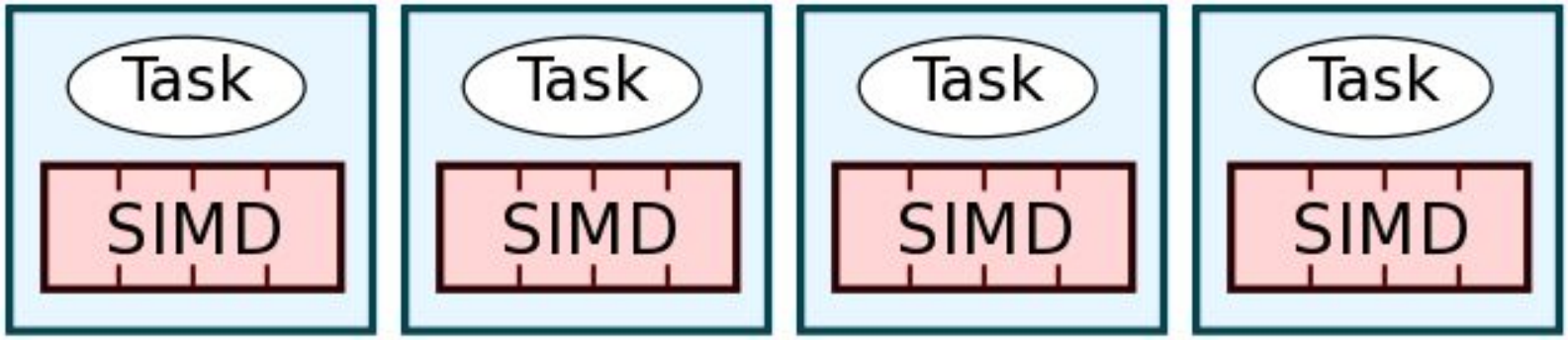


Intra-Core

- Packed-SIMD Vectorization
  - Intel AVX, Arm NEON

Inter-Core

- Task-Based Parallel Programming Frameworks
  - Intel TBB, Cilk

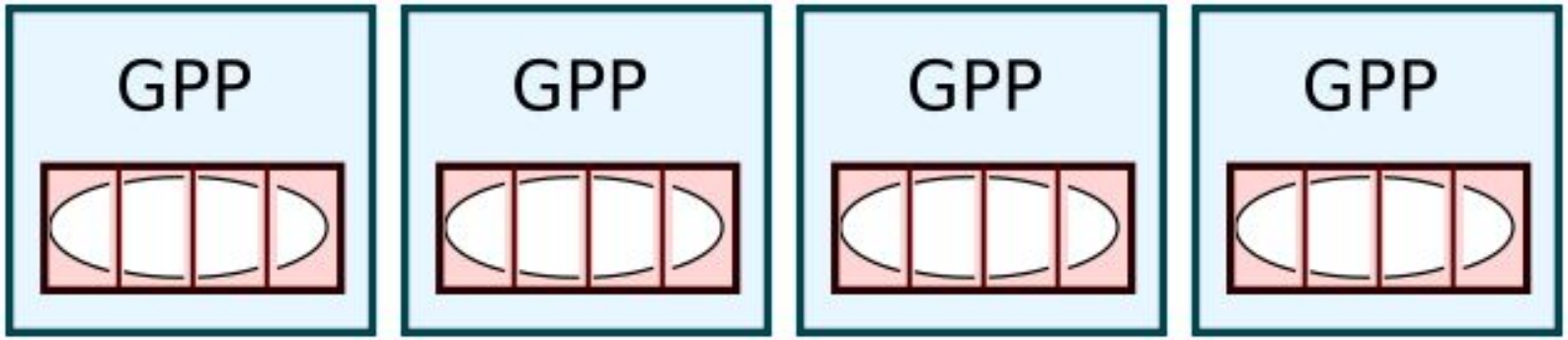


Intra-Core

- Packed-SIMD Vectorization
  - Intel AVX, Arm NEON

Inter-Core

- Task-Based Parallel Programming Frameworks
  - Intel TBB, Cilk



Intra-Core

- Packed-SIMD Vectorization
  - Intel AVX, Arm NEON

Challenges of Combining Tasks and Vectors

```
void app_kernel_tbb_avx(int N, float* src, float* dst) {
  // Pack data into padded aligned chunks
  //   src -> src_chunks[NUM_CHUNKS * SIMD_WIDTH]
  //   dst -> dst_chunks[NUM_CHUNKS * SIMD_WIDTH]
  ...

  // Use TBB across cores
  parallel_for (range(0, NUM_CHUNKS, TASK_SIZE), [&] (range r) {
    for (int i = r.begin(); i < r.end(); i++) {
      // Use packed-SIMD within a core
      #pragma simd vlen(SIMD_WIDTH)
      for (int j = 0; j < SIMD_WIDTH; j++) {
        if (src_chunks[i][j] > THRESHOLD)
          aligned_dst[i] = DoLightCompute(aligned_src[i]);
        else
          aligned_dst[i] = DoHeavyCompute(aligned_src[i]);
  ...
  ...
```

Challenge #1: Intra-Core Parallel Abstraction Gap

Challenges of Combining Tasks and Vectors

```
void app_kernel_tbb_avx(int N, float* src, float* dst) {
  // Pack data into padded aligned chunks
  //   src -> src_chunks[NUM_CHUNKS * SIMD_WIDTH]
  //   dst -> dst_chunks[NUM_CHUNKS * SIMD_WIDTH]
  ...

  // Use TBB across cores
  parallel_for (range(0, NUM_CHUNKS, TASK_SIZE), [&] (range r) {
    for (int i = r.begin(); i < r.end(); i++) {
      // Use packed-SIMD within a core
      #pragma simd vlen(SIMD_WIDTH)
      for (int j = 0; j < SIMD_WIDTH; j++) {
        if (src_chunks[i][j] > THRESHOLD)
          aligned_dst[i] = DoLightCompute(aligned_src[i]);
        else
          aligned_dst[i] = DoHeavyCompute(aligned_src[i]);
  ...
  ...
```

Challenge #1: Intra-Core Parallel Abstraction Gap

Challenges of Combining Tasks and Vectors

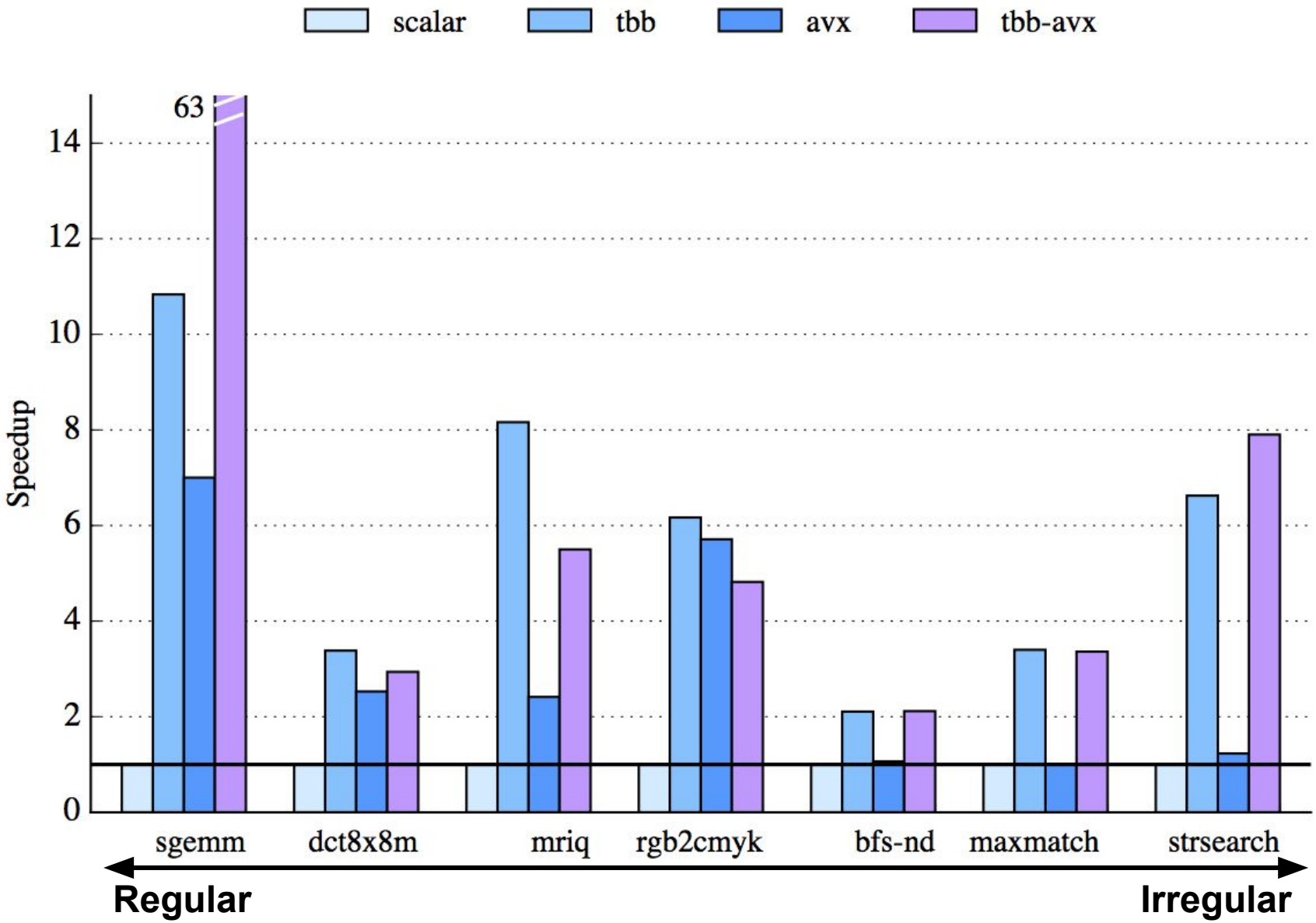
```
void app_kernel_tbb_avx(int N, float* src, float* dst) {
  // Pack data into padded aligned chunks
  //   src -> src_chunks[NUM_CHUNKS * SIMD_WIDTH]
  //   dst -> dst_chunks[NUM_CHUNKS * SIMD_WIDTH]
  ...

  // Use TBB across cores
  parallel_for (range(0, NUM_CHUNKS, TASK_SIZE), [&] (range r) {
    for (int i = r.begin(); i < r.end(); i++) {
      // Use packed-SIMD within a core
      #pragma simd vlen(SIMD_WIDTH)
      for (int j = 0; j < SIMD_WIDTH; j++) {
        if (src_chunks[i][j] > THRESHOLD)
          aligned_dst[i] = DoLightCompute(aligned_src[i]);
        else
          aligned_dst[i] = DoHeavyCompute(aligned_src[i]);
  ...
  ...
```

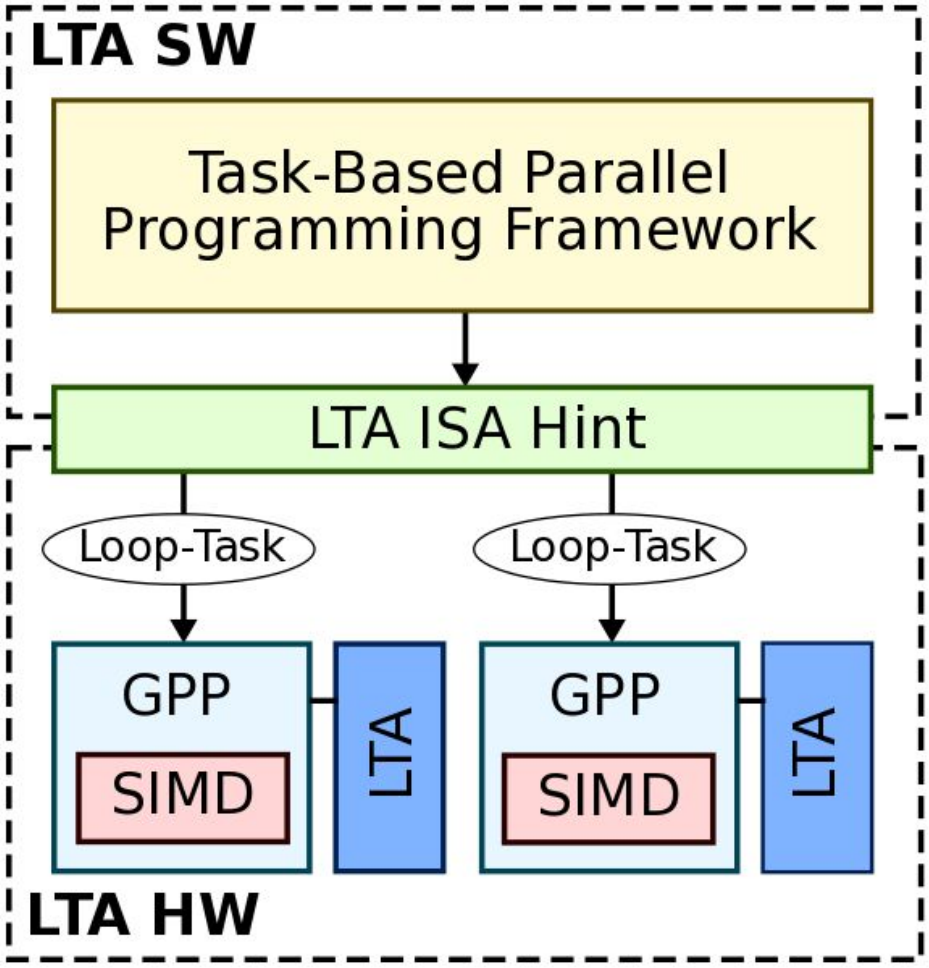
Challenge #1: Intra-Core Parallel Abstraction Gap

Challenge #2: Inefficient Execution of Irregular Tasks

Native Performance Results



Loop-Task Accelerator (LTA) Vision



- Motivation
- **Challenge #1: LTA SW**
- Challenge #2: LTA HW
- Evaluation
- Conclusion

LTA SW: API and ISA Hint

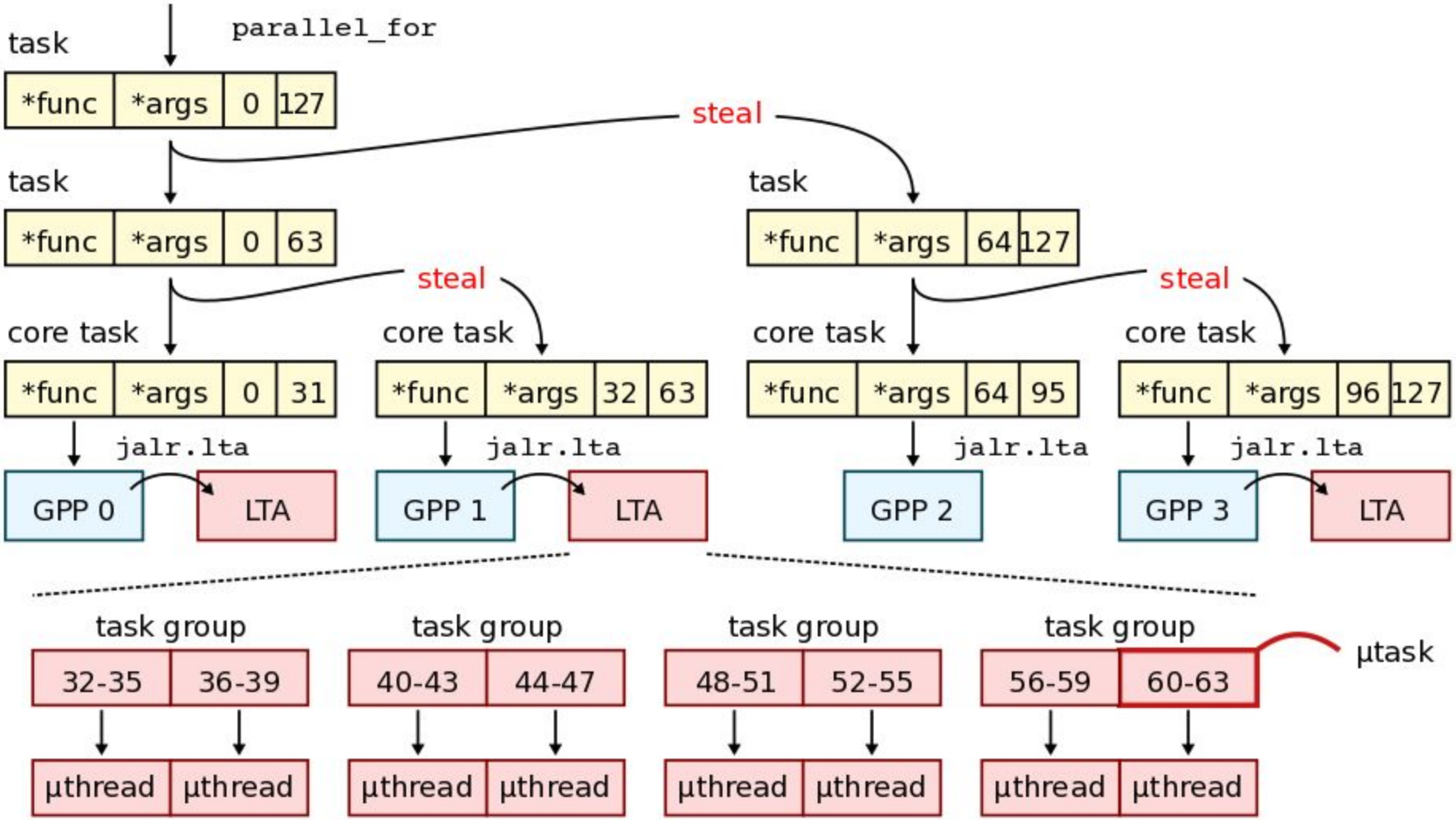
```
void app_kernel_lta(int N, float* src, float* dst) {
  LTA_PARALLEL_FOR(0, N, (dst, src), ({
    if (src[i] > THRESHOLD)
      dst[i] = DoComputeLight(src[i]);
    else
      dst[i] = DoComputeHeavy(src[i]);
  }));
}

void loop_task_func(void* a, int start, int end, int step=1);
```

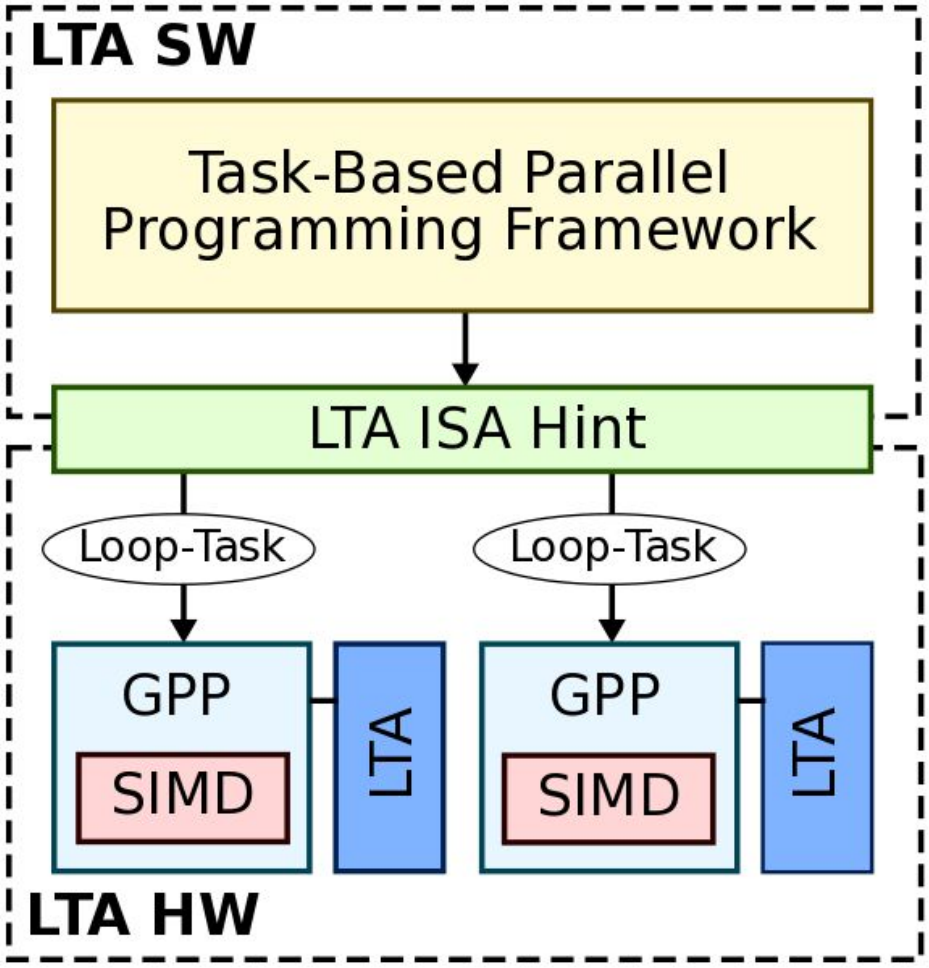
```
jalr.lta $rd, $rs
```

Hint that hardware can potentially accelerate task execution

LTA SW: Task-Based Runtime

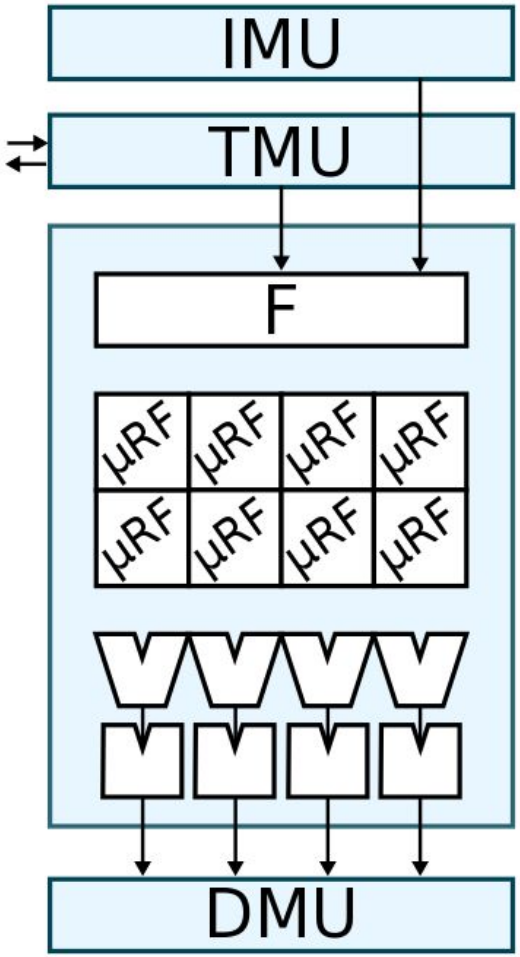


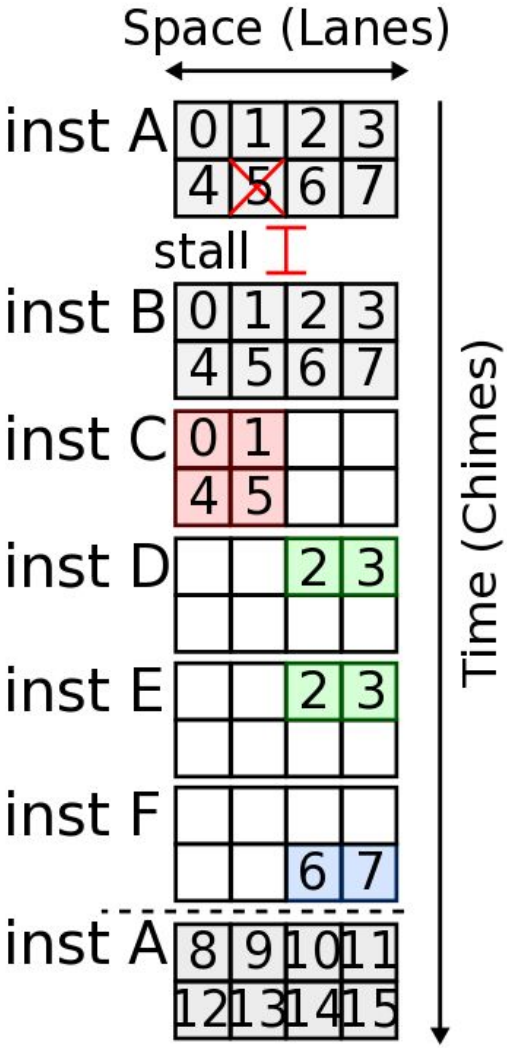
Loop-Task Accelerator (LTA) Vision



- Motivation
- Challenge #1: LTA SW
- **Challenge #2: LTA HW**
- Evaluation
- Conclusion

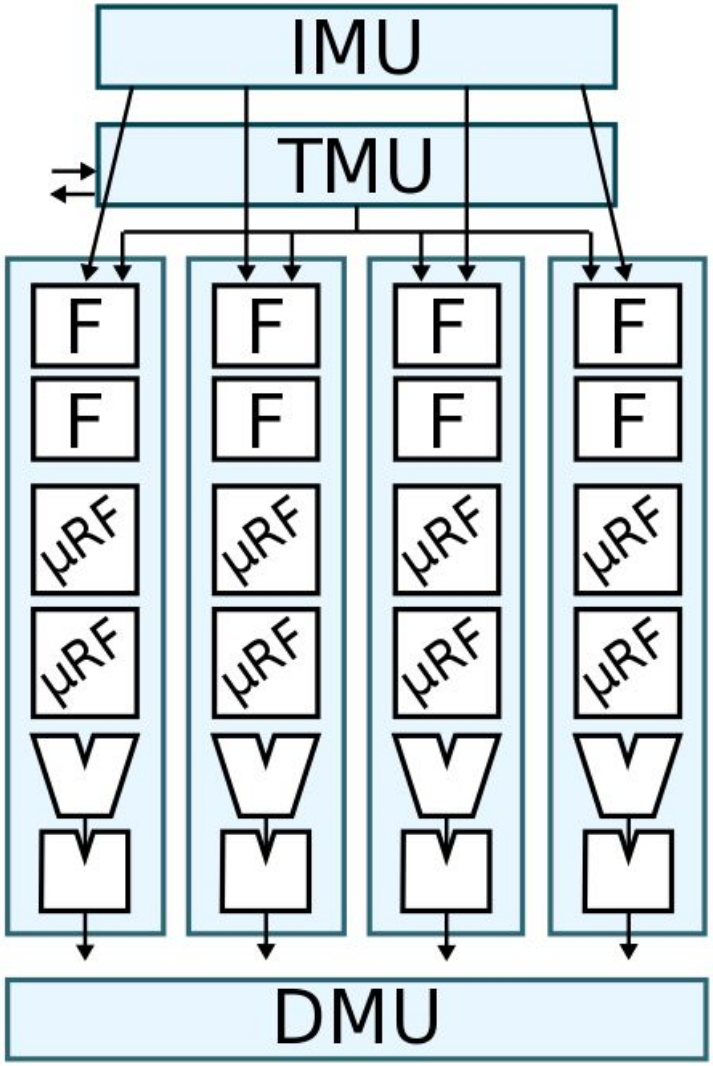
LTA HW: Fully-Coupled LTA

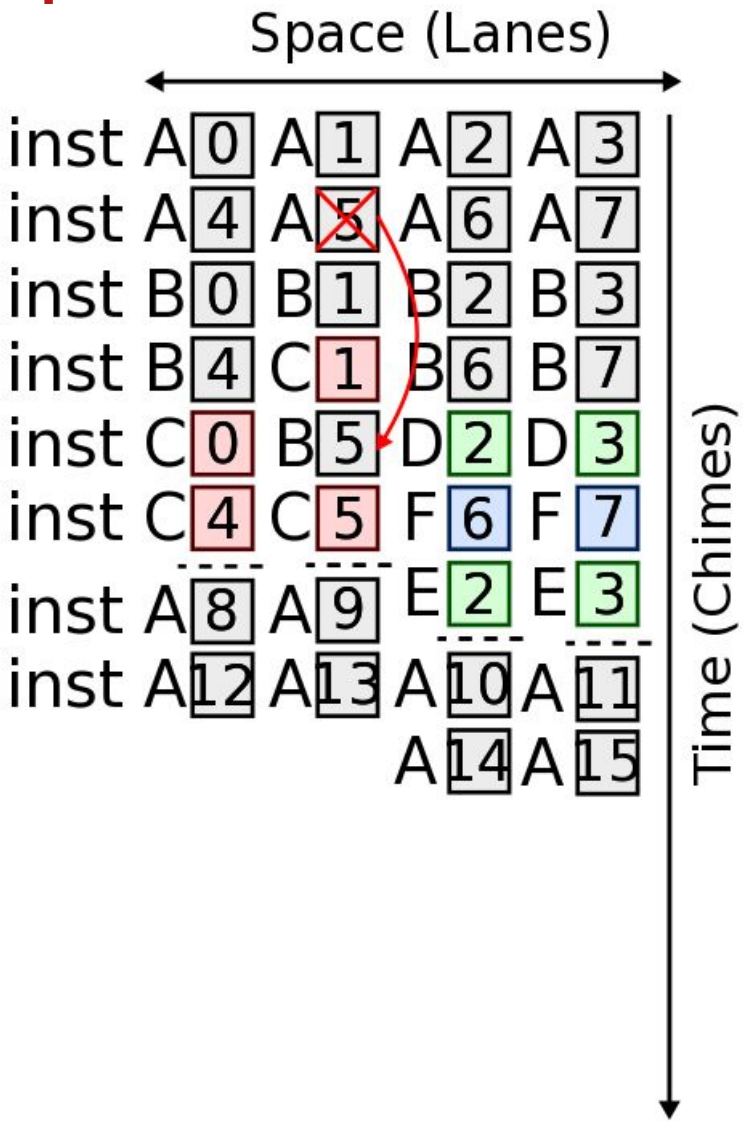




Coupling better for regular workloads (amortize frontend/memory)

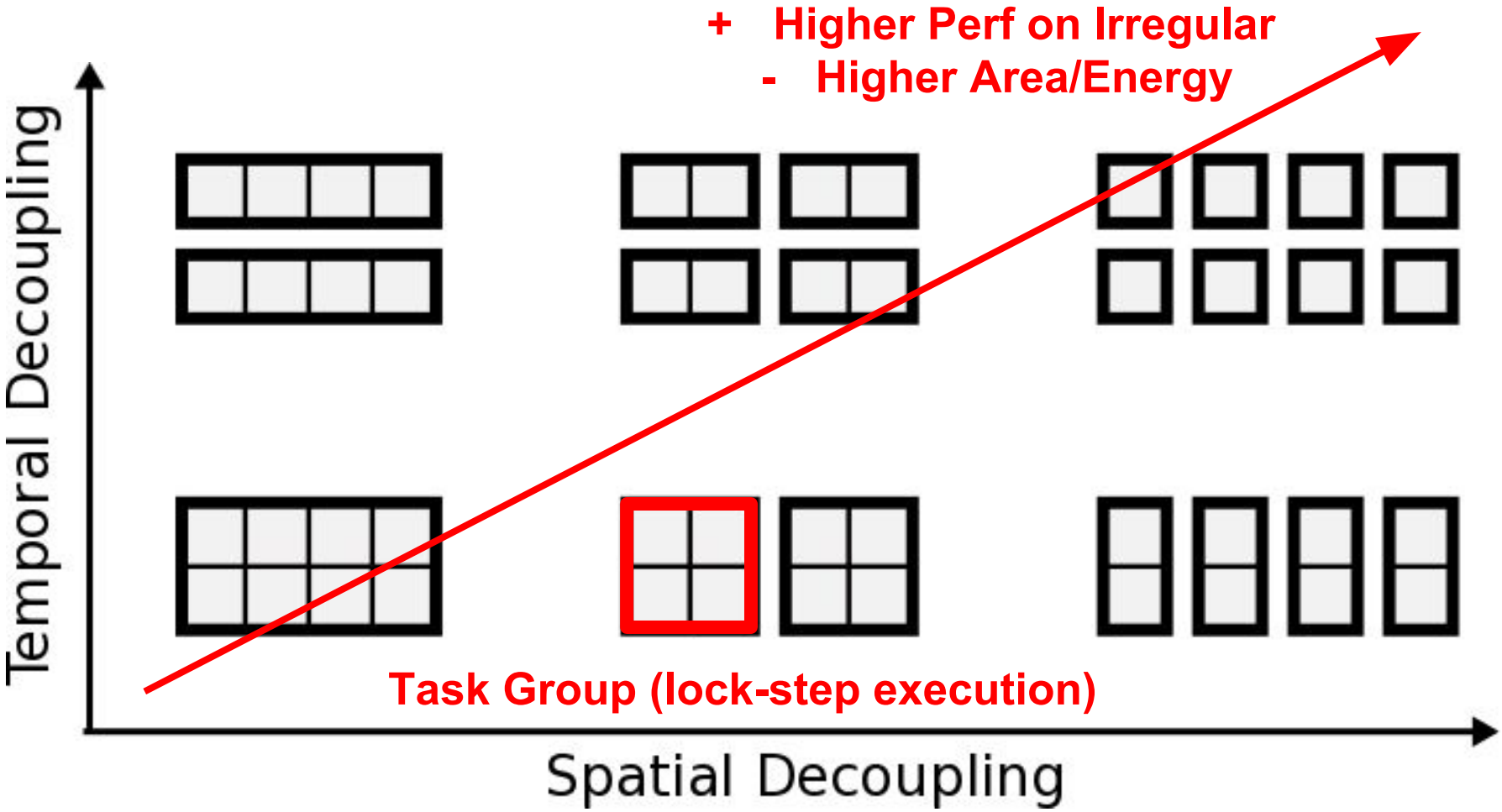
LTA HW: Fully Decoupled LTA





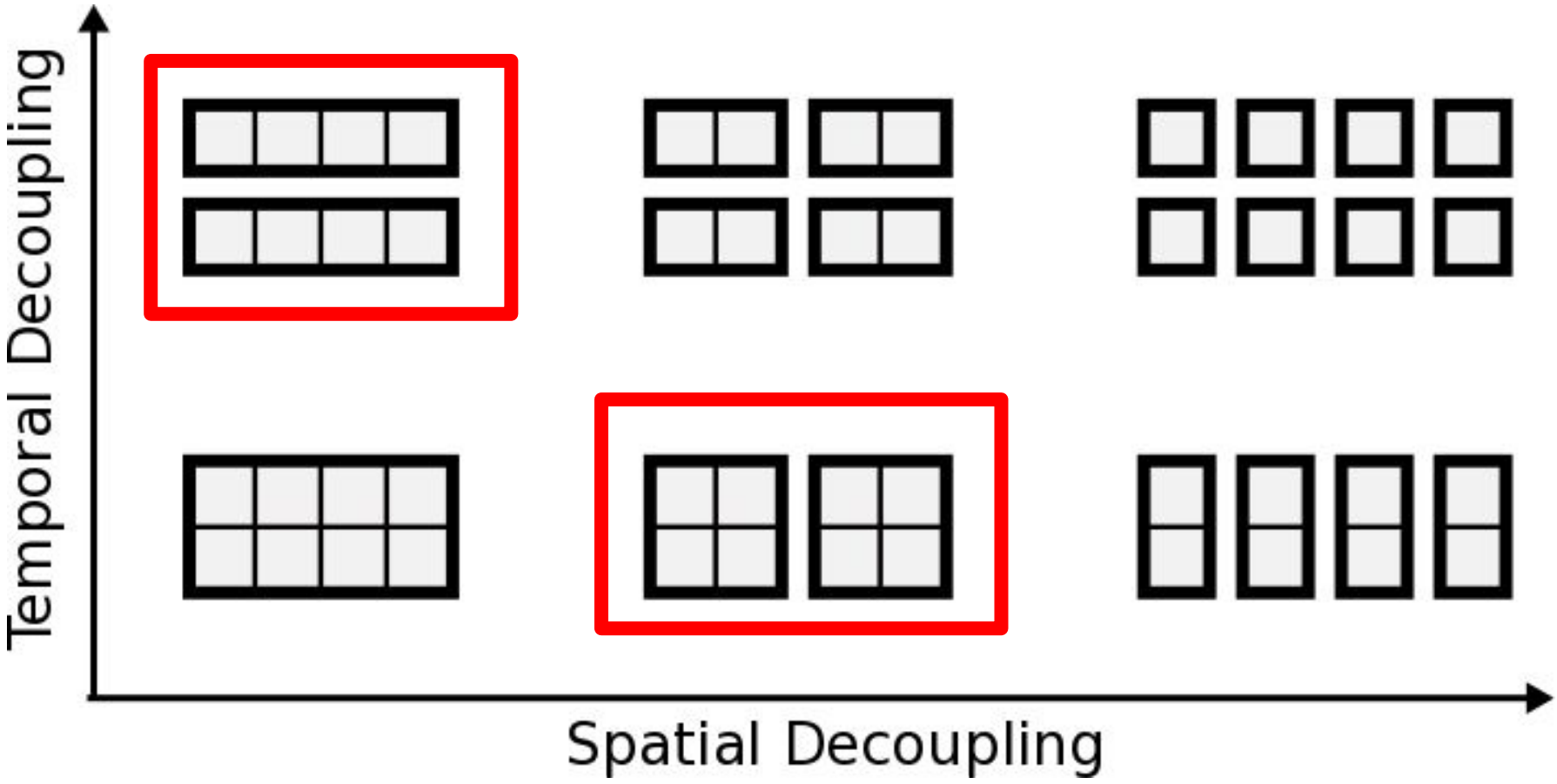
Decoupling better for irregular workloads (hide latencies)

LTA HW: Task-Coupling Taxonomy



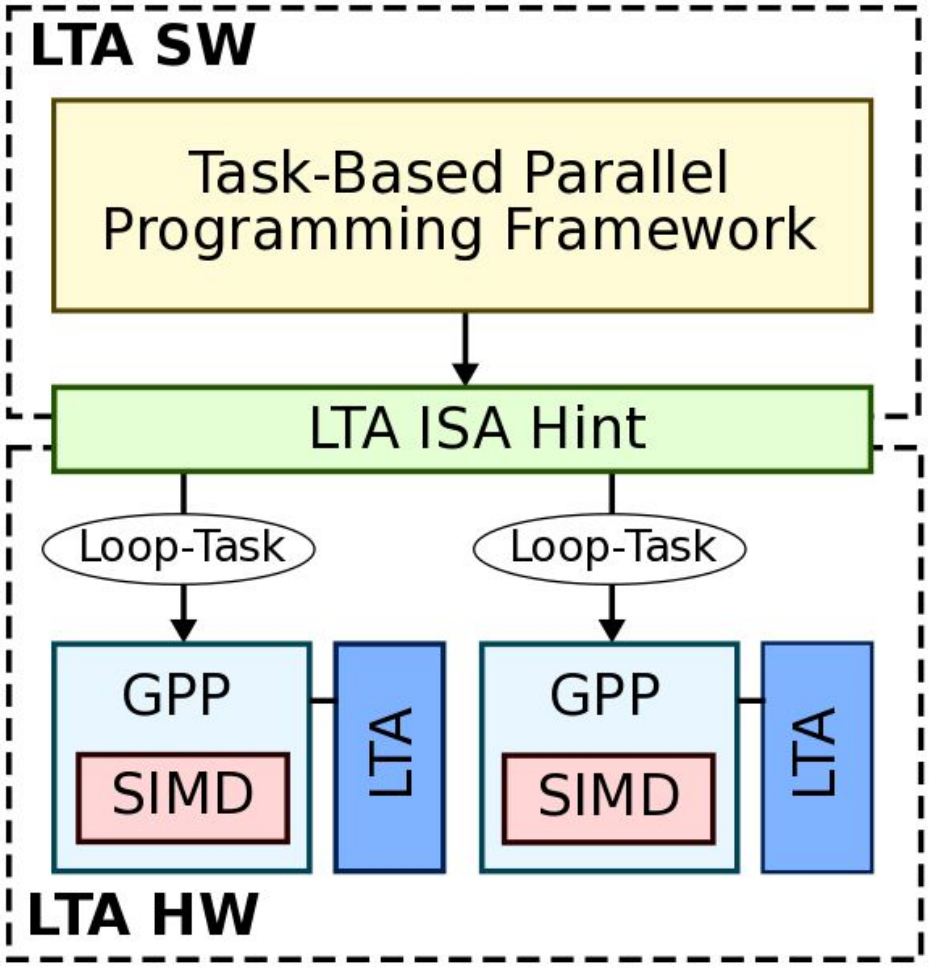
More decoupling (more task groups) in either space or time improves performance on irregular workloads at the cost of area/energy

LTA HW: Task-Coupling Taxonomy



Does it matter whether we decouple in space or in time?

Loop-Task Accelerator (LTA) Vision

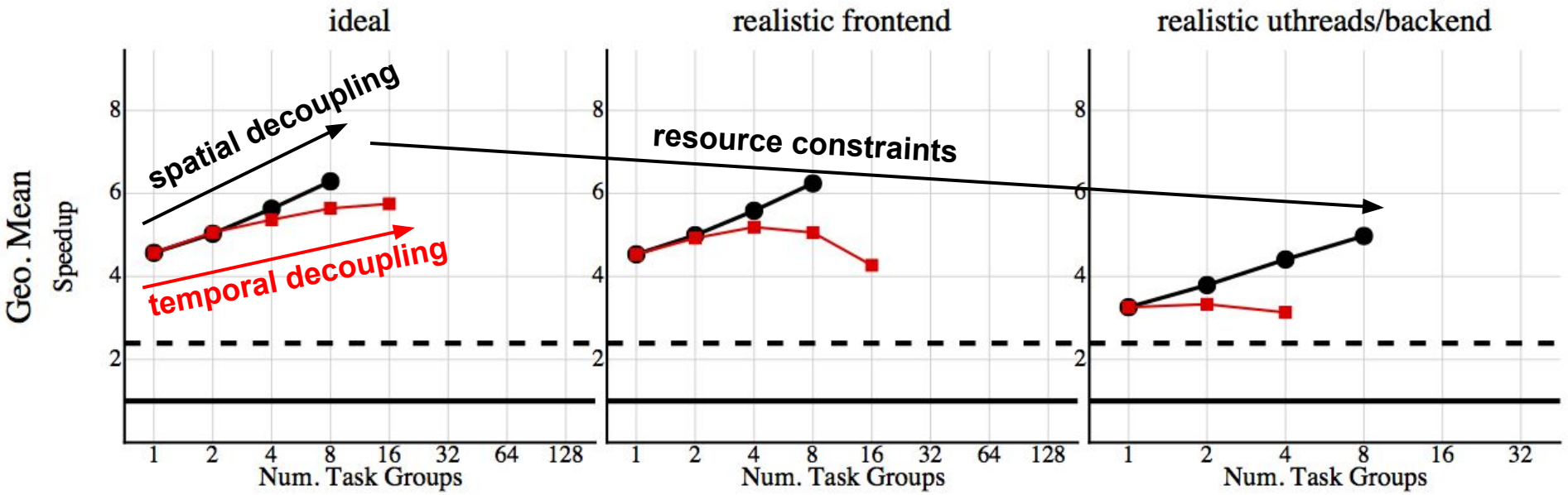


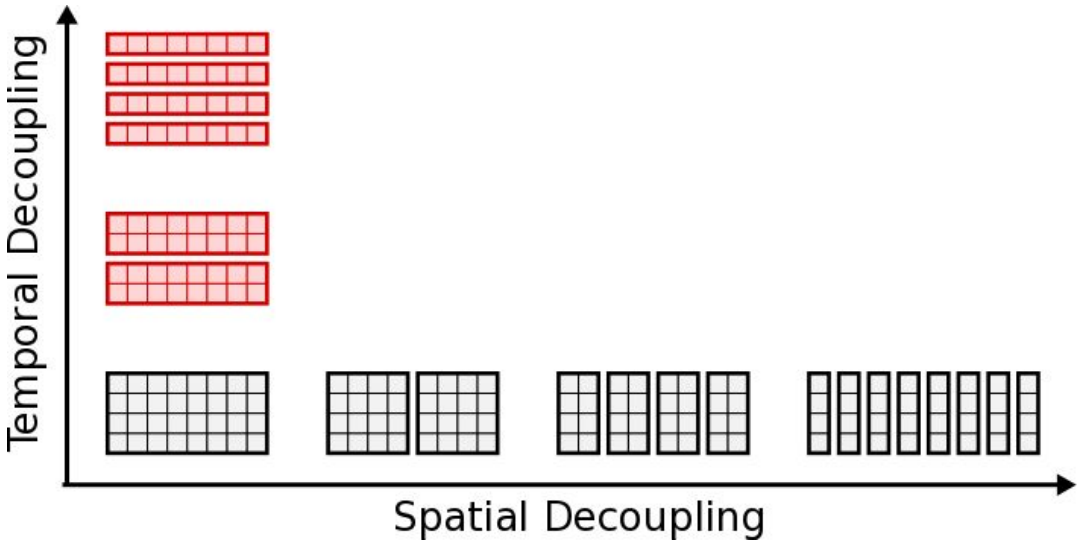
- Motivation
- Challenge #1: LTA SW
- Challenge #2: LTA HW
- **Evaluation**
- Conclusion

Evaluation: Methodology

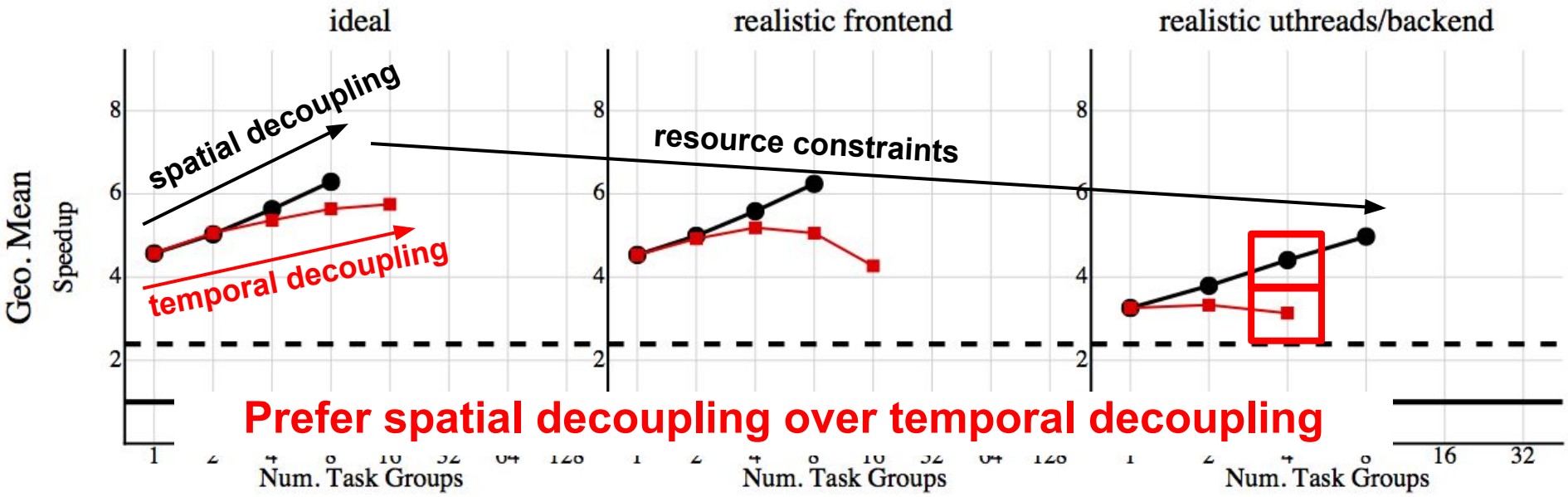
- Ported 16 application kernels from PBBS and in-house benchmark suites with diverse loop-task parallelism
  - Scientific computing: N-body simulation, MRI-Q, SGEMM
  - Image processing: bilateral filter, RGB-to-CMYK, DCT
  - Graph algorithms: breadth-first search, maximal matching
  - Search/Sort algorithms: radix sort, substring matching
- gem5 + PyMTL co-simulation for cycle-level performance
- Component/event-based area/energy modeling
  - Uses area/energy dictionary backed by VLSI results and McPAT

Evaluation: Design Space Exploration

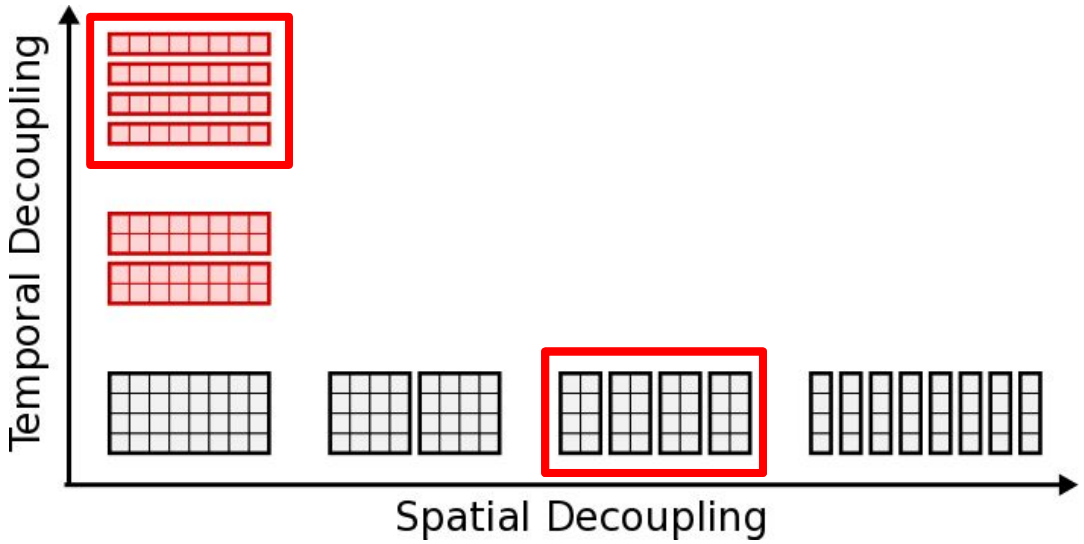




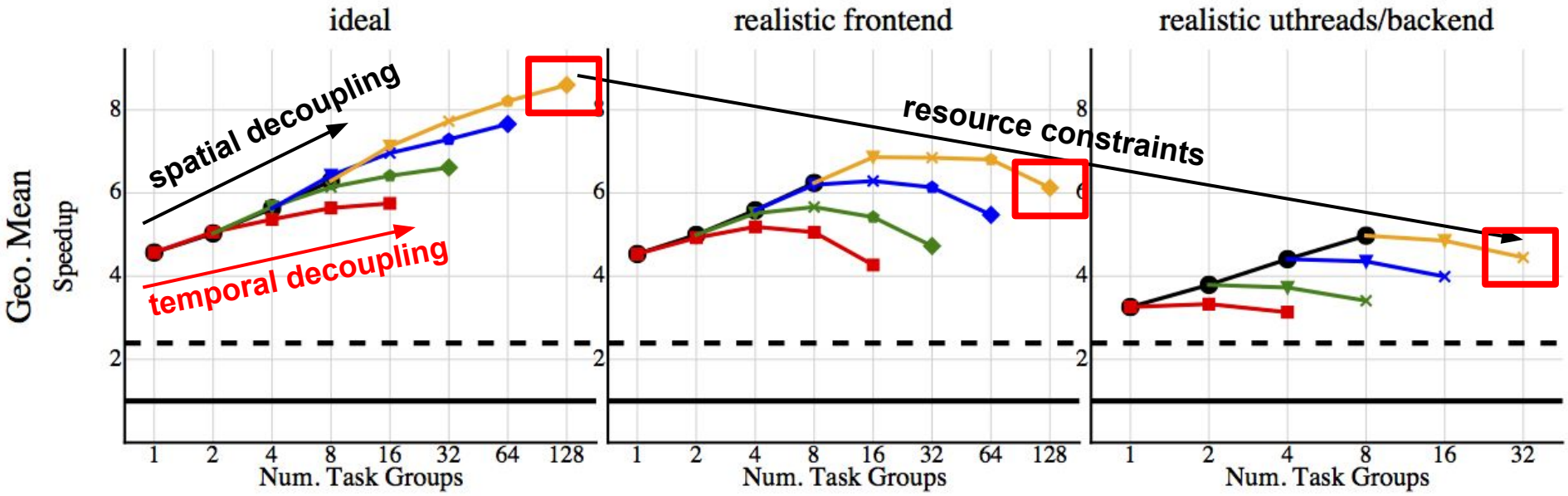
Evaluation: Design Space Exploration

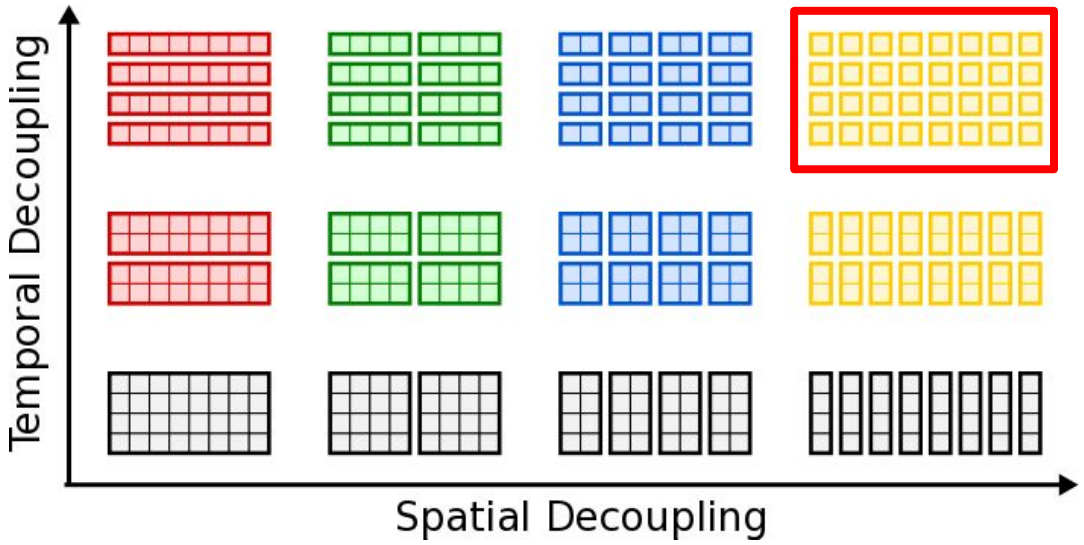


Prefer spatial decoupling over temporal decoupling

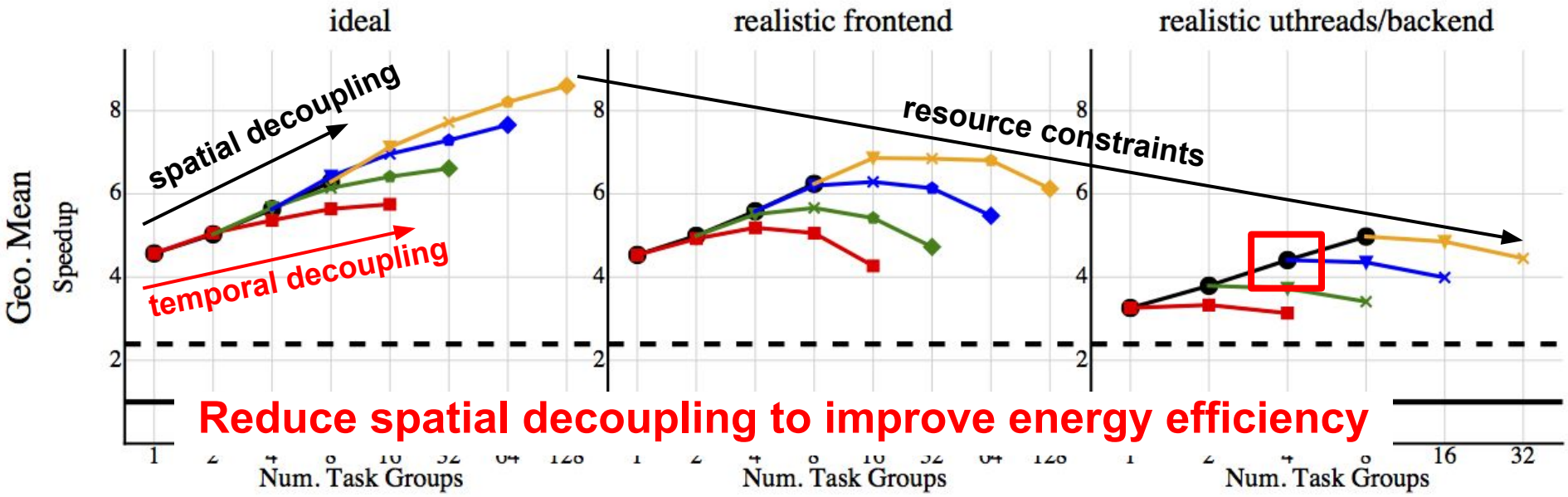


Evaluation: Design Space Exploration

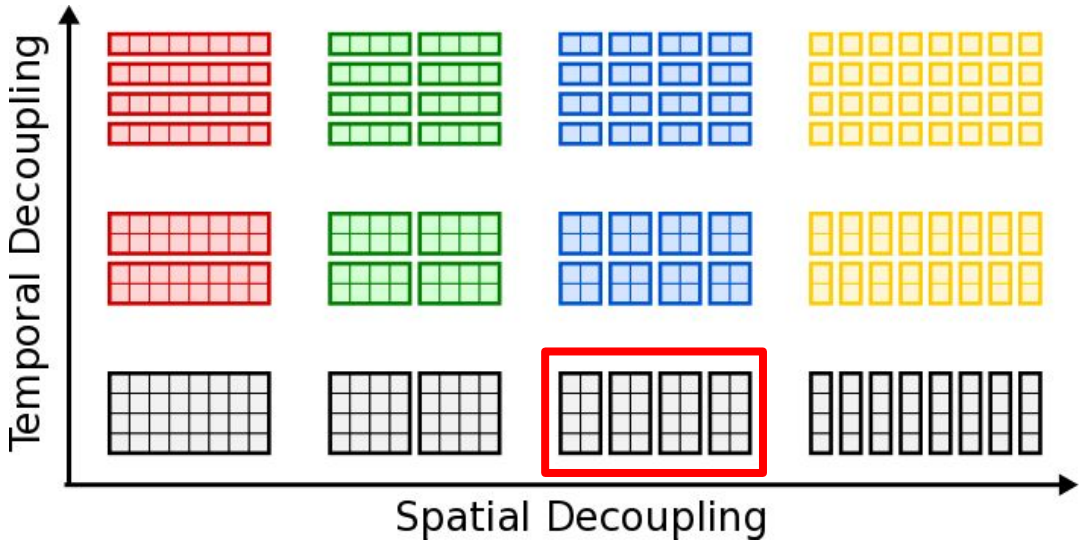




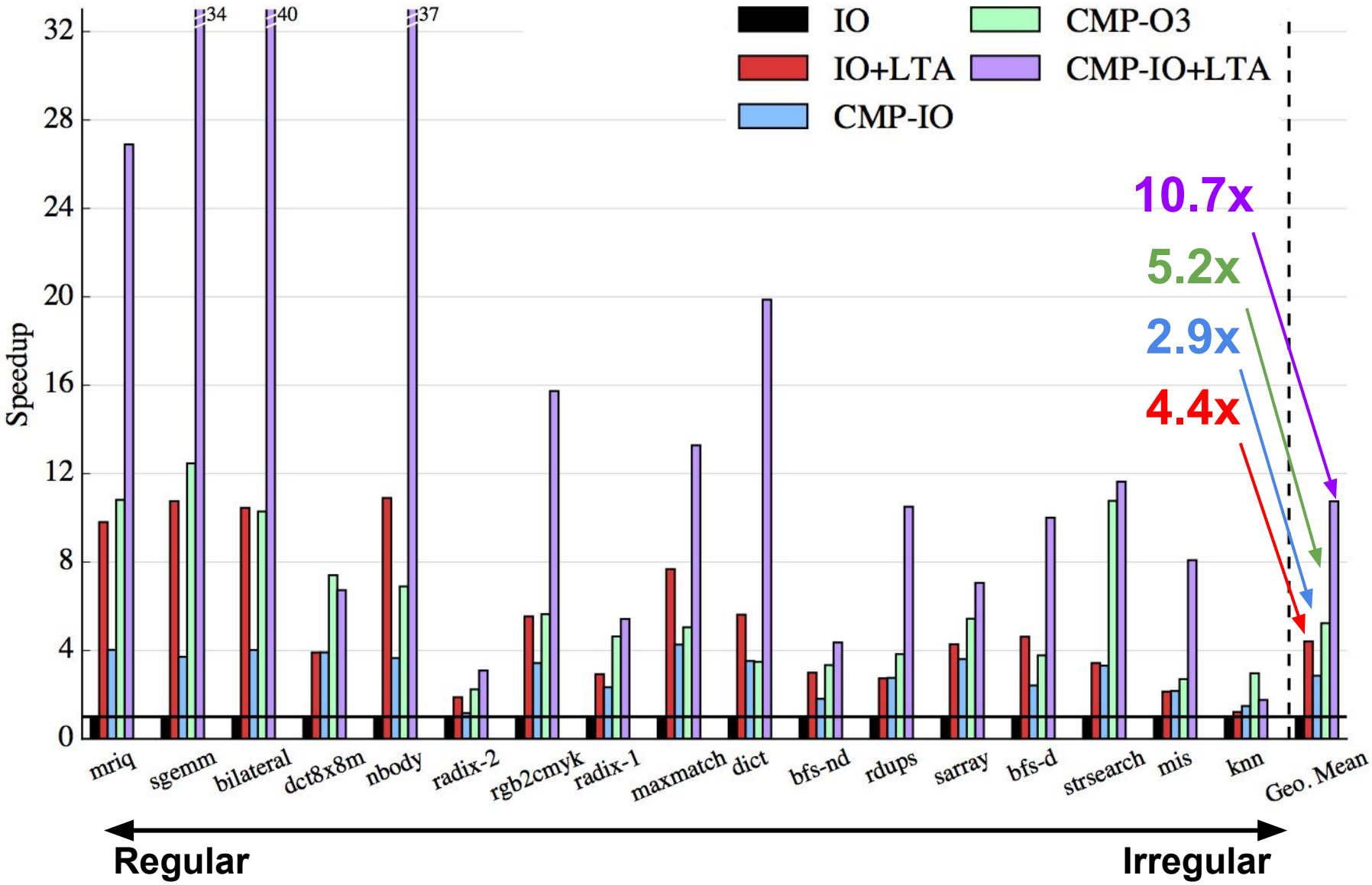
Evaluation: Design Space Exploration



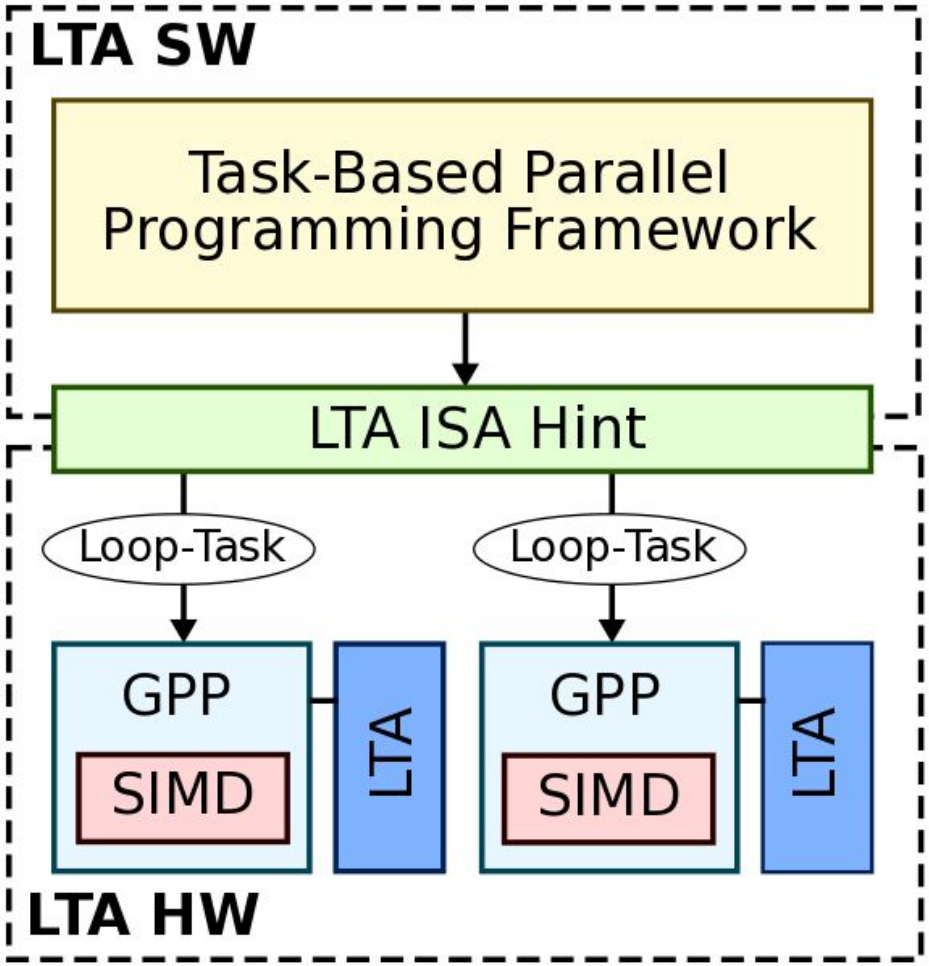
Reduce spatial decoupling to improve energy efficiency

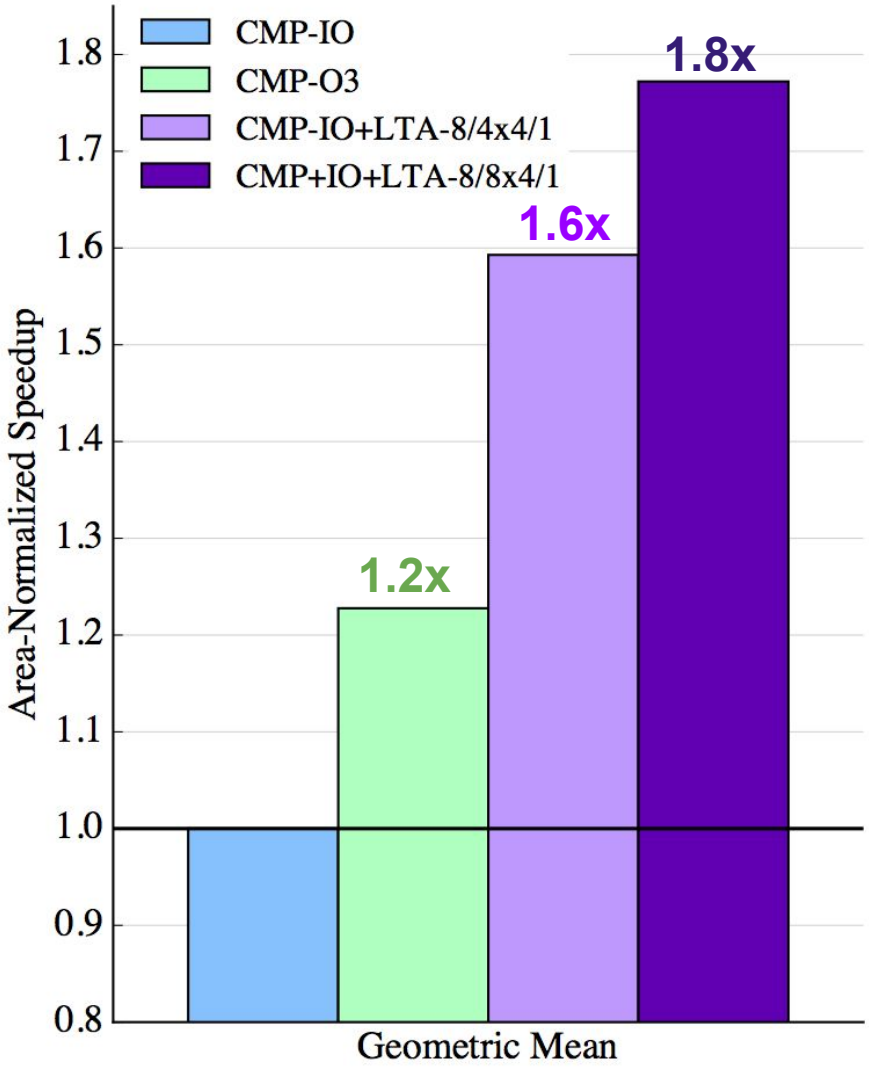


Evaluation: Multicore LTA Performance

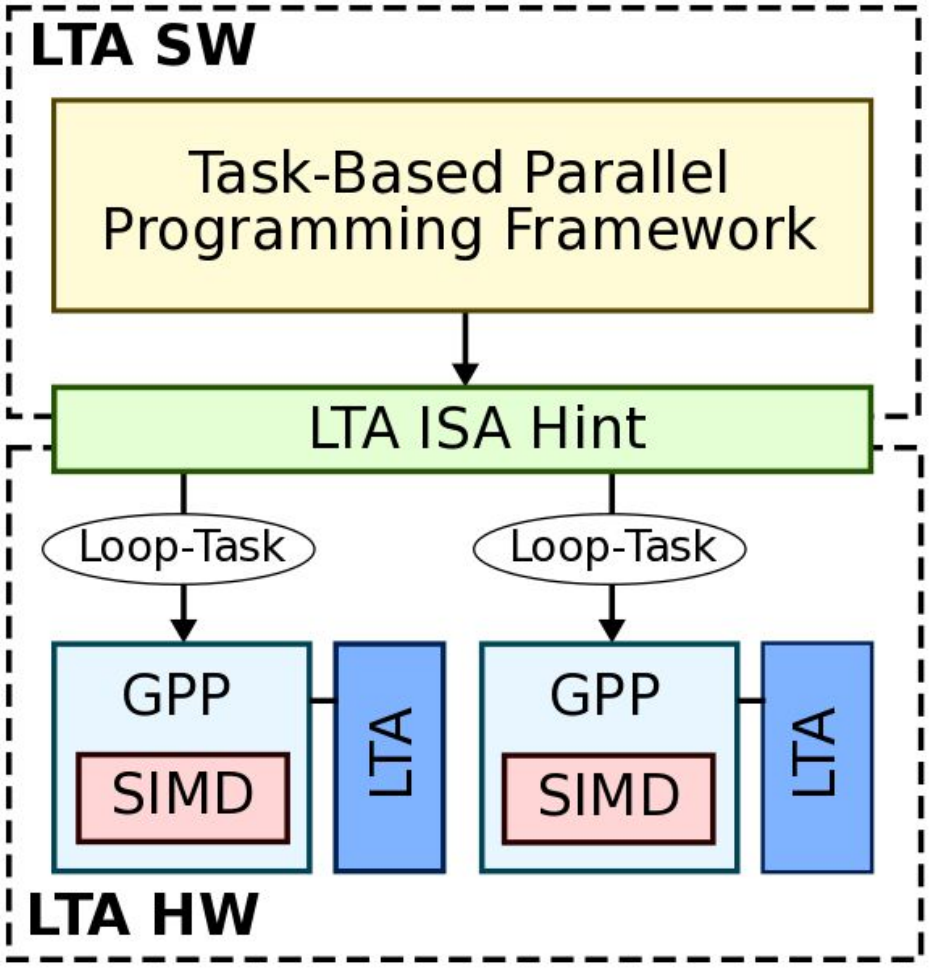


Evaluation: Area-Normalized Performance





Loop-Task Accelerator (LTA) Vision



- Motivation
- Challenge #1: LTA SW
- Challenge #2: LTA HW
- Evaluation
- **Conclusion**

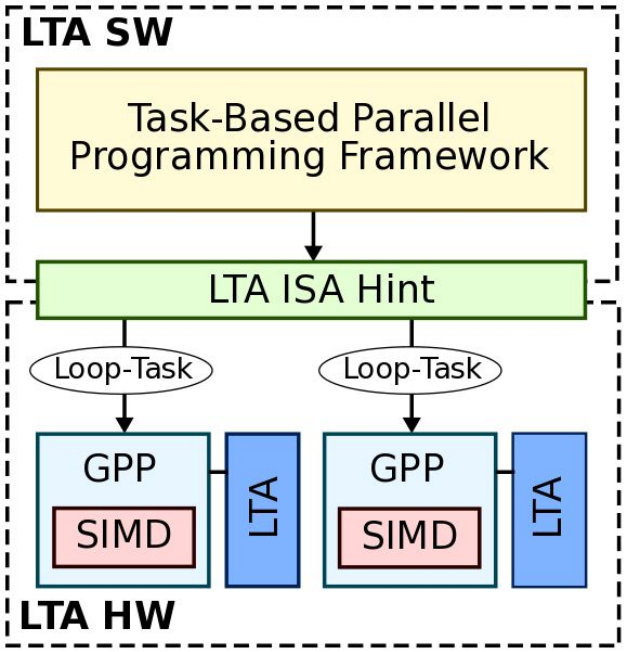
Related Work

- **Challenge #1: Intra-Core Parallel Abstraction Gap**
  - Persistent threads for GPGPUs (S. Tzeng et al.)
  - OpenCL, OpenMP, C++ AMP
  - Cilk for vectorization (B. Ren et al.)
  - And more...

- **Challenge #2: Inefficient Execution of Irregular Tasks**
  - Variable warp sizing (T. Rogers et al.)
  - Temporal SIMT (S. Keckler et al.)
  - Vector-lane threading (S. Rivoire et al.)
  - And more…

- Please see paper for more detailed references!

Take-Away Points



- **Intra-core parallel abstraction gap** and **inefficient execution of irregular tasks** are fundamental challenges for CMPs
- LTAs address both challenges with **a lightweight ISA hint and a flexible microarchitectural template**
- Results suggest in a resource-constrained environment, architects should favor **spatial decoupling over temporal decoupling**
- First step towards accelerating a wider variety of task parallelism