PAGANI: A PARALLEL ADAPTIVE GPU ALGORITHM FOR NUMERICAL INTEGRATION

Authors: Ioannis Sakiotis, Kamesh Arumugam, Marc Paterno, Desh Ranjan, Balsa Terzic, Mohammad Zubair

NVIDIA

OLD DOMINION UNIVERSITY

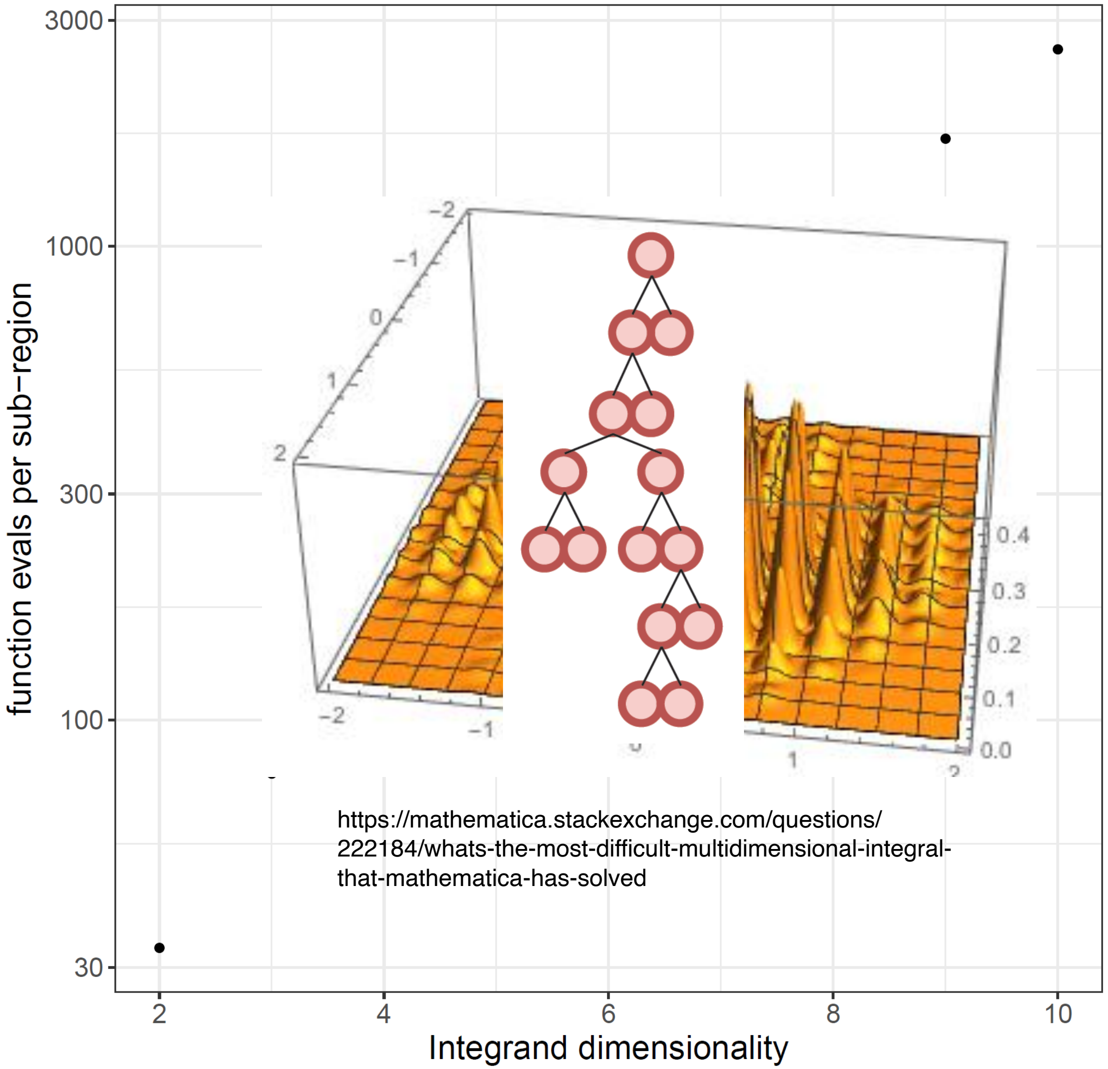
Fermilab

INTRODUCTION

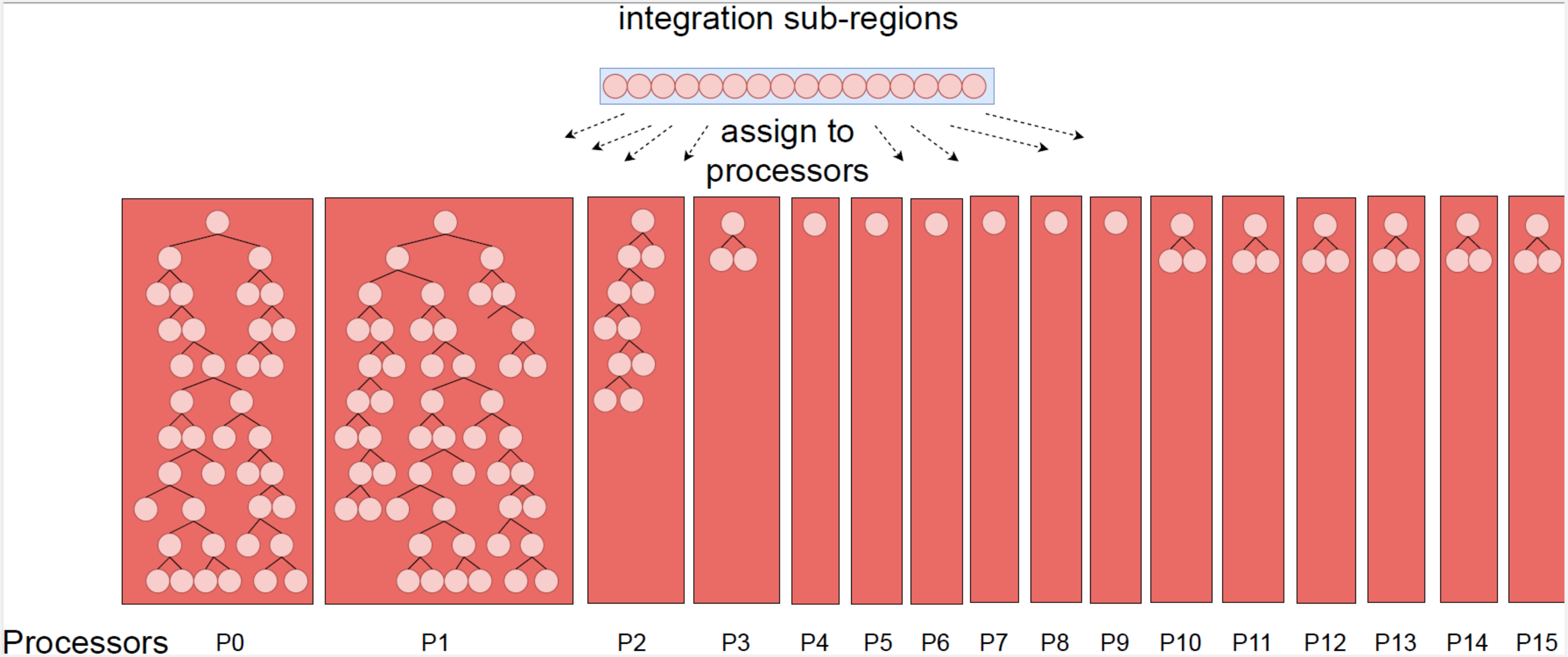
- Applications: parameter estimation, simulation of beam dynamics, risk management, ray-tracing
- Probabilistic, deterministic
- No algorithm guarantees the accuracy of its estimated results
- Evaluate accuracy
- Specify desired accuracy (relative error tolerance, digits of precision, etc.)
- Goal: Bring adaptive quadrature to GPUs

BACKGROUND: QUADRATURE

- Weighted summation $\sum (w_i\ f(x_i))$
- Error-estimate
- Expect large initial error
- Apply weighted summation in sub-region to reduce error
- Highly parallelizable
- Number of points grows exponentially with the number of dimensions
- Uniform split and sub-region evaluation infeasible
- Cuhre
  - $2^n + \Theta(n^3)$ functions evaluations (n-dimensions)
  - Attractive option for low/mid dimensional integrands
  - Priority-queue

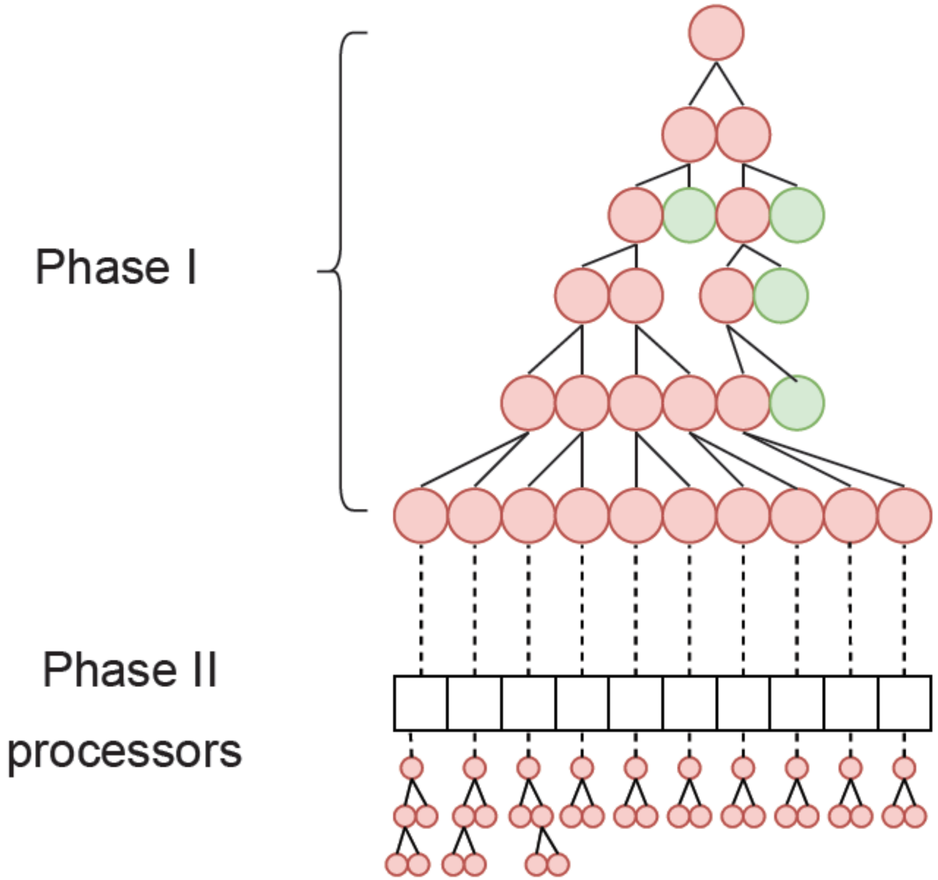


NAÏVE PARALLELIZATION



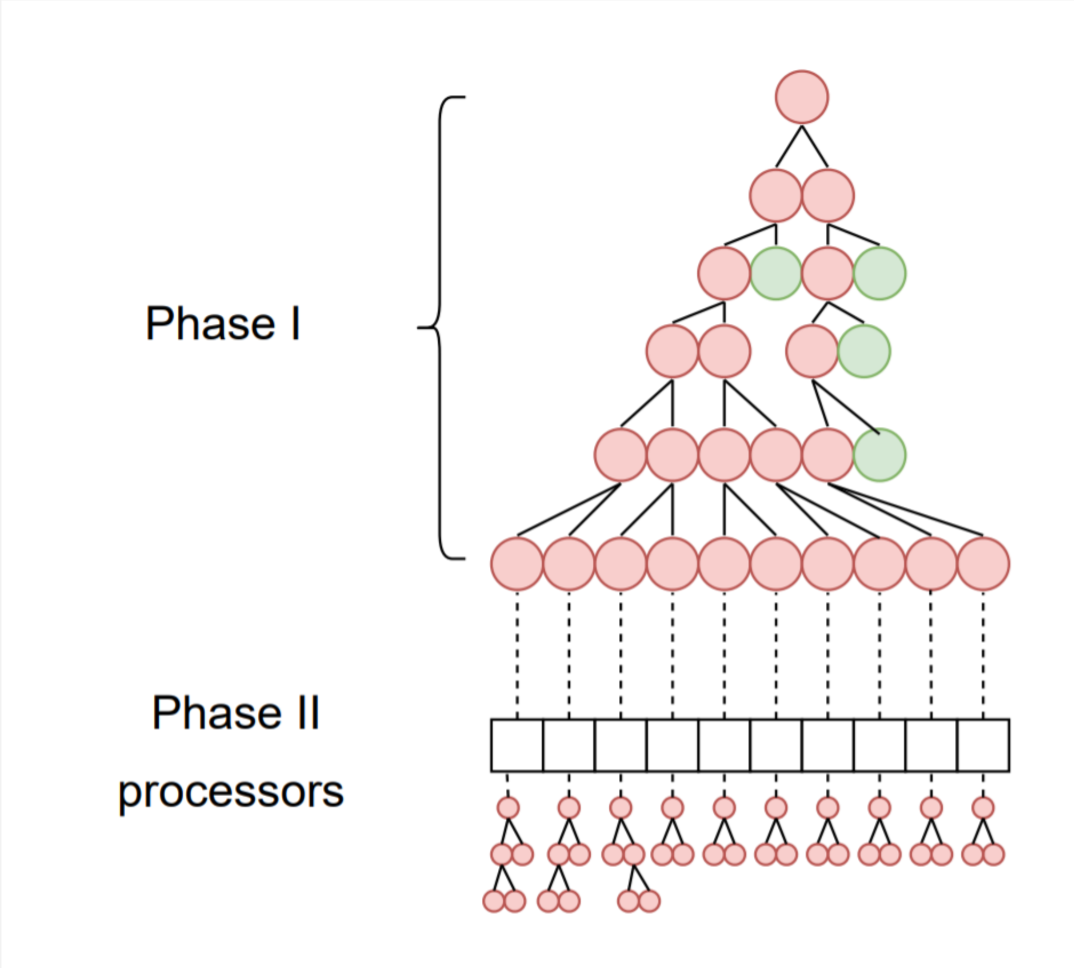
BACKGROUND: TWO-PHASE CUHRE

- GPU-targeted algorithm
- Assign each sub-region to a processor
- **Applies the Cuhre algorithm on each sub-region in parallel**
- Utilizes pre-processing Phase 1
  - Generate sufficiently large workload (# of sub-regions)
  - Load-balancing
- No synchronization between processors
- Local termination



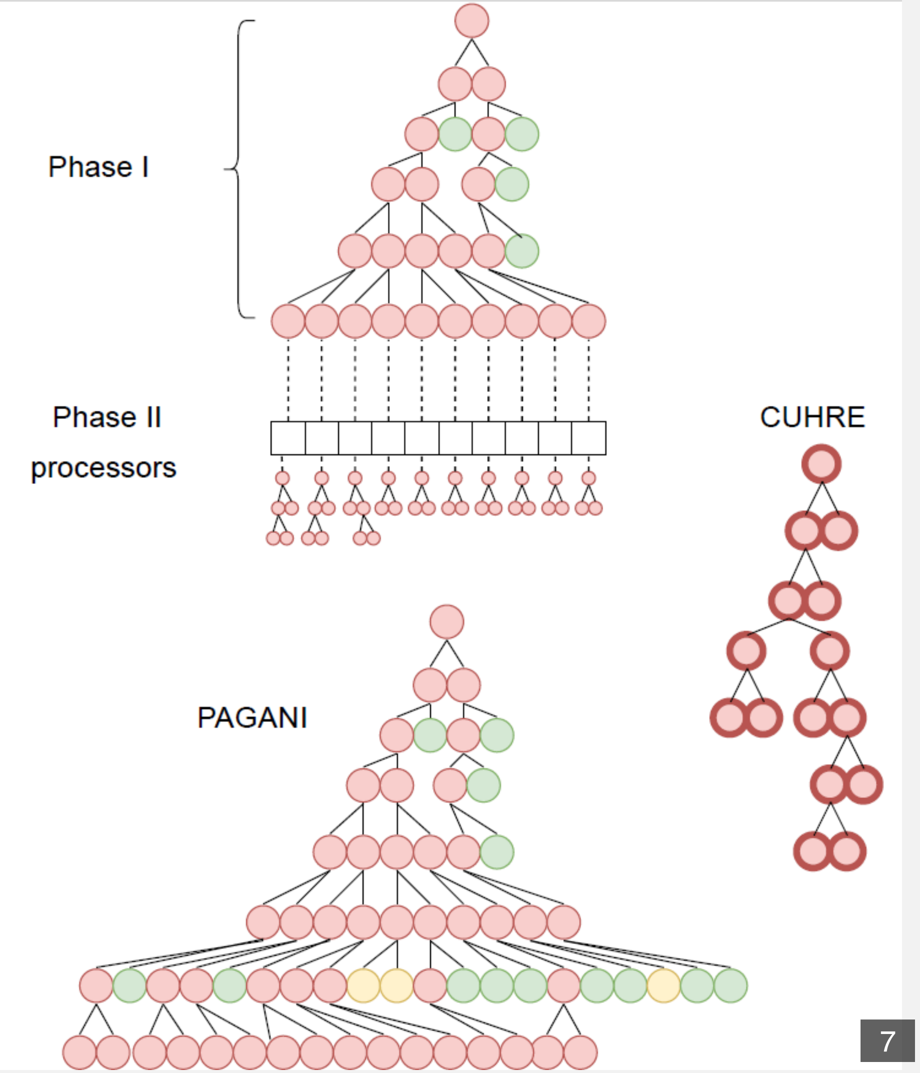
TWO-PHASE CUHRE

- Utilization of sequential algorithm by parallel processors
- Poor load-balancing when high-precision
- Local data-structures
- Unknown global state (unless global sync.)
- Local termination



PAGANI

- Parallel algorithm designed for massively parallel architectures
- Avoid sequential scheme
- Sub-divide all sub-regions
- Filtering instead of sorting
- Green/yellow = accurate enough
- Avoid synchronization after isolated processing
- Uniform workload
- Bound by memory



ALGORITHM DESCRIPTION

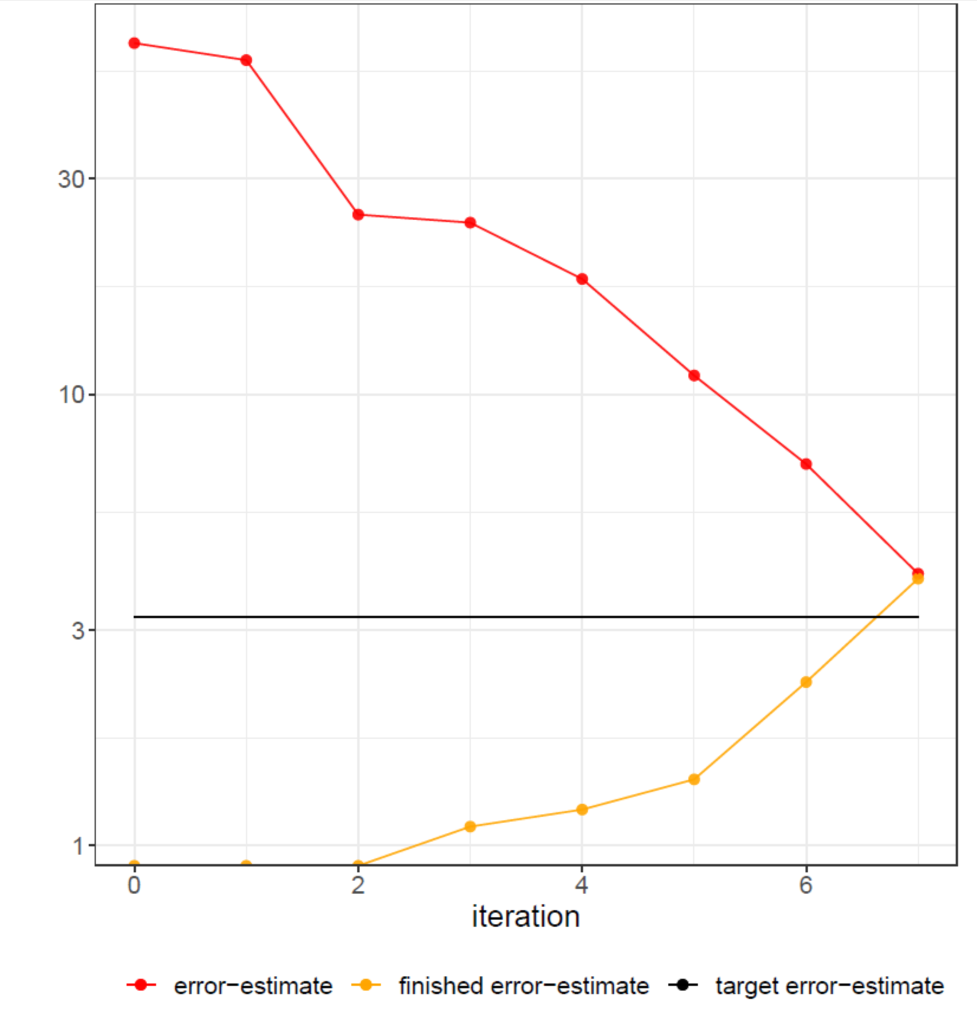
- Initial uniform-split
- Parallel sub-region evaluation
- Two-level error-estimate
- Relative error classification
  - Finished/active
- Summations
- Termination conditions
- Conditional threshold classification
- Filtering
- Split all active regions

Algorithm 2 PAGANI Algorithm

procedure PAGANI(f, n, $b[n]$, τ_{rel}, τ_{abs})
 $R_0 \leftarrow b$
 $s \leftarrow d^n$ ▷ region list size
 $H \leftarrow$ Uniform-Split(R_0, d)
 $A[1:s] \leftarrow 1$
 $V[1:s], E[1:L], K[1:s] \leftarrow 0$
 $V_p[1:s], E_p[1:s] \leftarrow 0$
 $v, e, v_f, e_f \leftarrow 0$ ▷ cumulative/finished estimates
 for $it : it_{max}$ **do**
  $V, E, K \leftarrow$ Evaluate(H)
  $E < -$ Two-Level-Error(V, E, V_p, E_p)
  $A < -$ Rel-Err-Classify(V, E, A)
  $v \leftarrow$ Sum(V)
  $e \leftarrow$ Sum(E)
  if $\frac{e+e_f}{|v+v_f|} \leq \tau_{rel}$ or $e + e_f \leq \tau_{abs}$ **then**
   return $v + v_f, e + e_f$
  $A \leftarrow$ Threshold-Classify ($A, E, v + v_f, e + e_f, v, e, s$)
  $v_f \leftarrow v-$ Sum($V \cdot A$) $+v_f$
  $e_f \leftarrow e-$ Sum($E \cdot A$) $+e_f$
  $H, V, E, L \leftarrow$ Filter(H, V, E, A)
  $V_p \leftarrow V, E_p \leftarrow E$ ▷ update all parents
  $H \leftarrow$ Split(H, K)
  $s \leftarrow 2s$

SUB-REGION CLASSIFICATION

- Why classify finished/active?
  - Only keep active regions in memory
  - Use relative error for global termination and finished/active classification
  - Regions with small estimates may not satisfy relative error termination
  - Remove regions that don't contribute "significantly"
- Aggressive filtering
  - How to define "significantly"?
  - Finished regions irrecoverable for performance
  - Balance finished and active estimates
  - Pick a threshold (initially the average) and adapt until criteria are met
  - Criteria: conserved memory, finished vs. active ratio
  - Perform if memory exhaustion or convergence of significant digits



INTEGRAND TEST SUITE

- First six integrands represent challenging integrand families
- Product and corner peaks, oscillatory, Gaussian, etc.
- Typically, randomized parameters
- Fixed parameters

$$f_1(x) = \cos\left(\sum_{i=1}^{8} i\, x_i\right)$$

$$f_2(x) = \prod_{i=1}^{6}\left(\frac{1}{50^2} + (x_i - 1/2)^2\right)^{-1}$$

$$f_3(x) = \left(1 + \sum_{i=1}^{d} i\, x_i\right)^{-d-1}$$

$$f_4(x) = \exp\left(-625 \sum_{i=1}^{d} (x_i - 1/2)^2\right)$$

$$f_5(x) = \exp\left(-10 \sum_{i=1}^{d} |x_i - 1/2|\right)$$

$$f_6(x) = \begin{cases} \exp\left(\sum_{i=1}^{6} (i+4)\, x_i\right) & \text{if } x_i < (3+i)/10 \\ 0 & \text{otherwise} \end{cases}$$

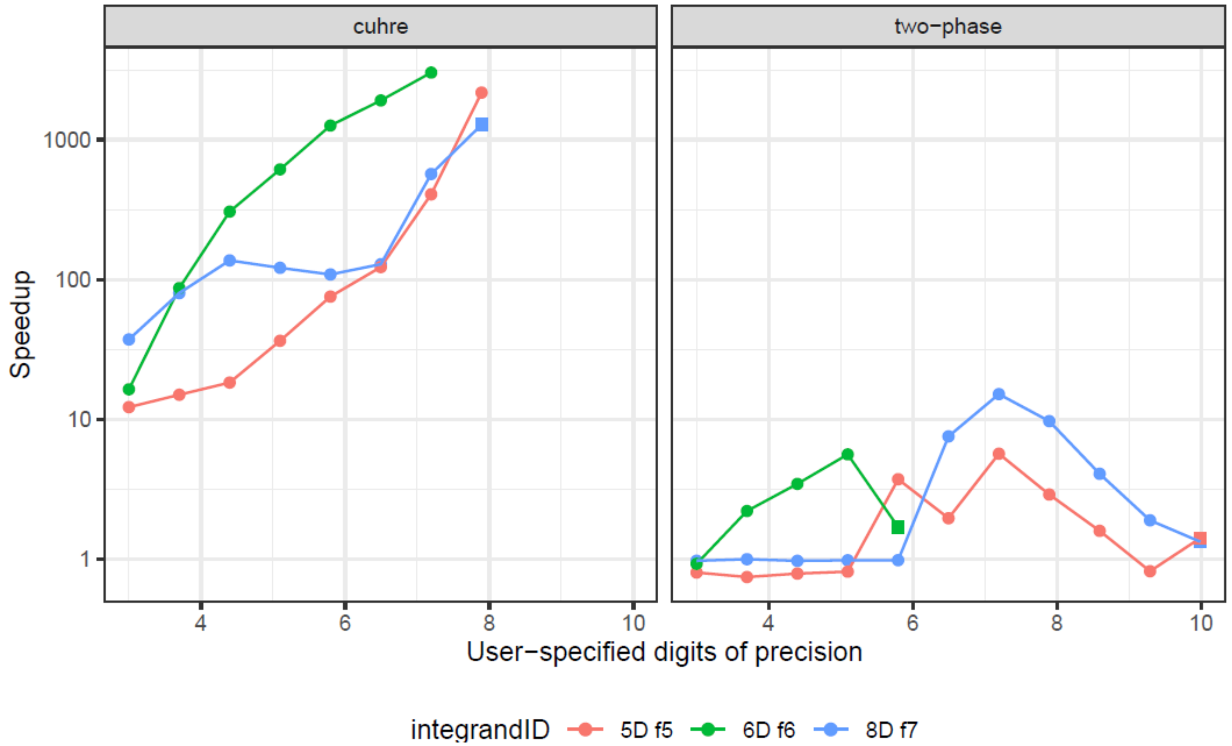
$$f_7(x) = \left(\sum_{i=1}^{d} x_i^2\right)^{11}$$

$$f_8(x) = \left(\sum_{i=1}^{d} x_i^2\right)^{15/2}$$

EXPERIMENTAL RESULTS

3 digits of precision = 1e-3 relative-error tolerance
4 digits of precision = 1e-4 relative-error tolerance

- CUDA 11 implementation
- Executed on V100 16 GB
- 2.4 GHz Xeon R Gold 6130 CPU
- Orders of magnitude speedup over sequential Cuhre
- Improved robustness over two-phase
  - Improved load-balancing on higher precision
  - More reliable error-estimate
- Comparable performance on low-precision

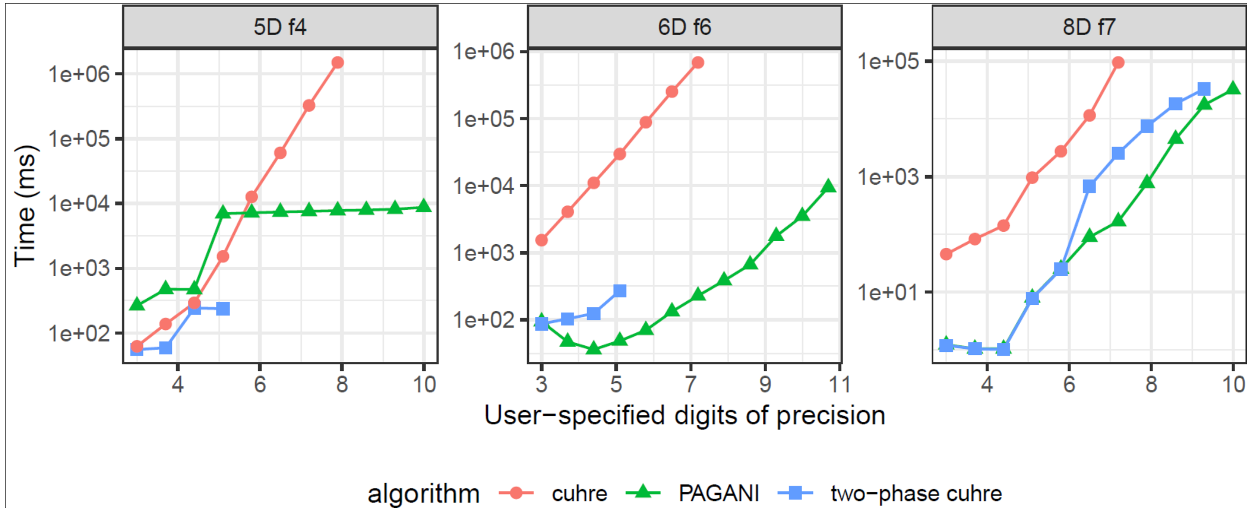


PERFORMANCE

Pagani generates more sub-regions

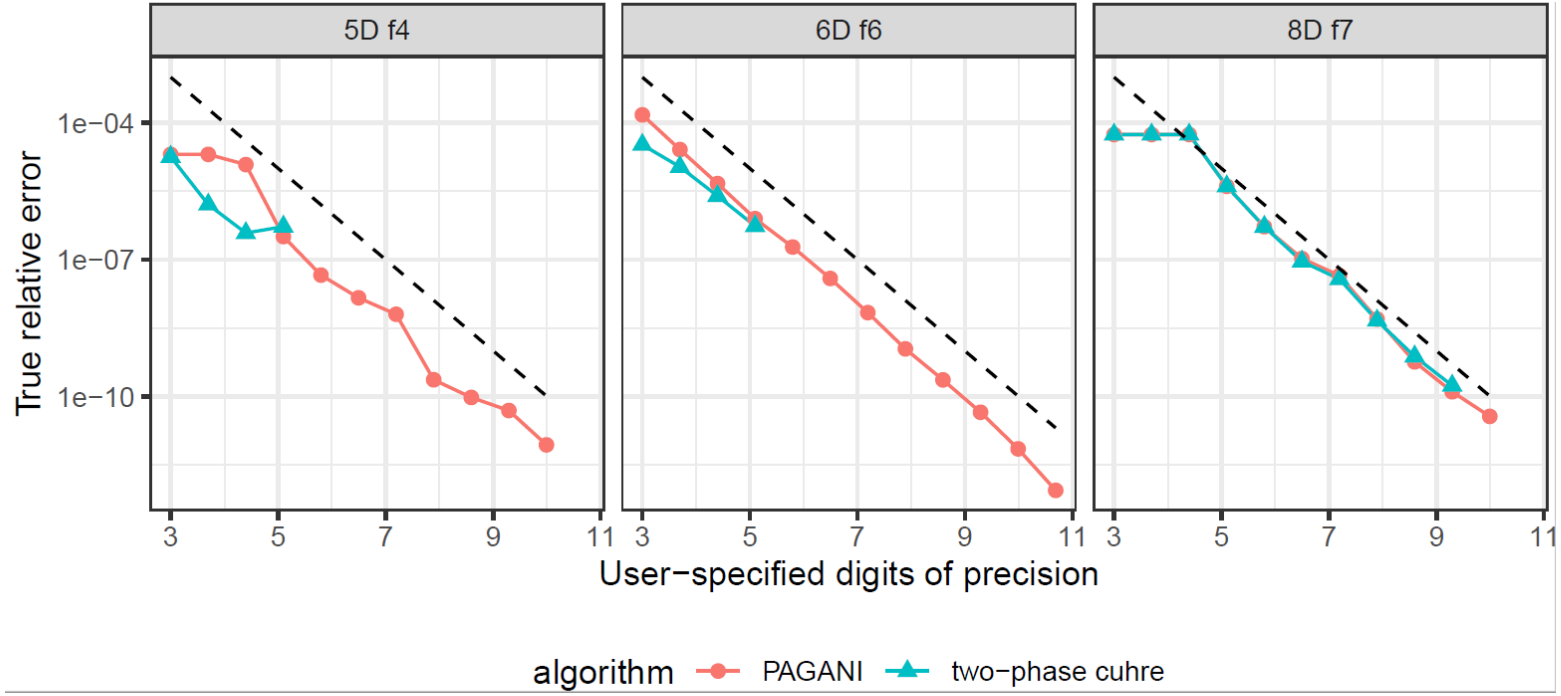
“Easy” integrals not worth the overhead

More digits-of-precision



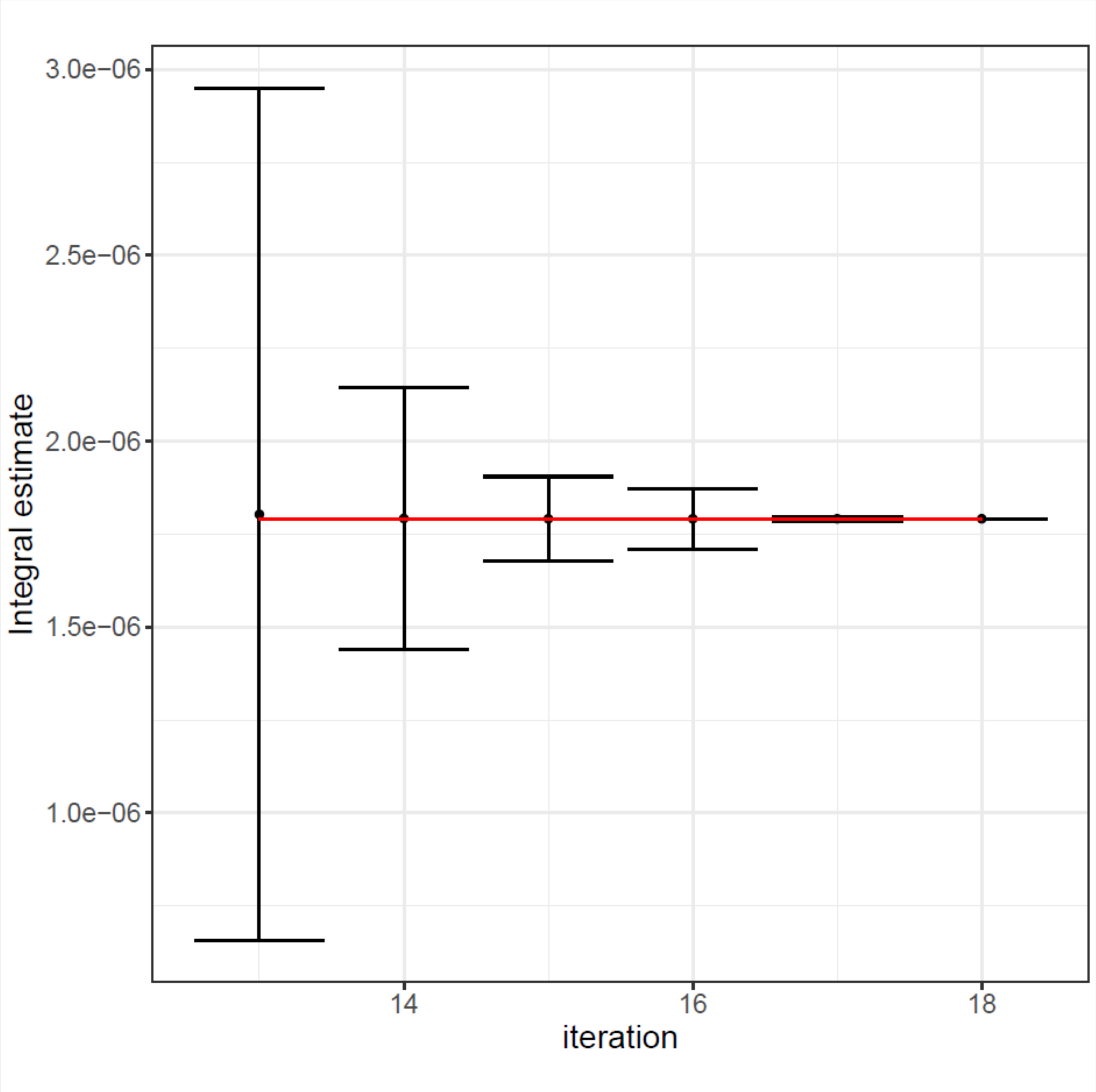
Accuracy

- No randomized parameters
- Evaluate error-estimate
- Claimed vs true relative-error
- Proximity to line indicates sub-division efficiency
- Check bounds of error estimate



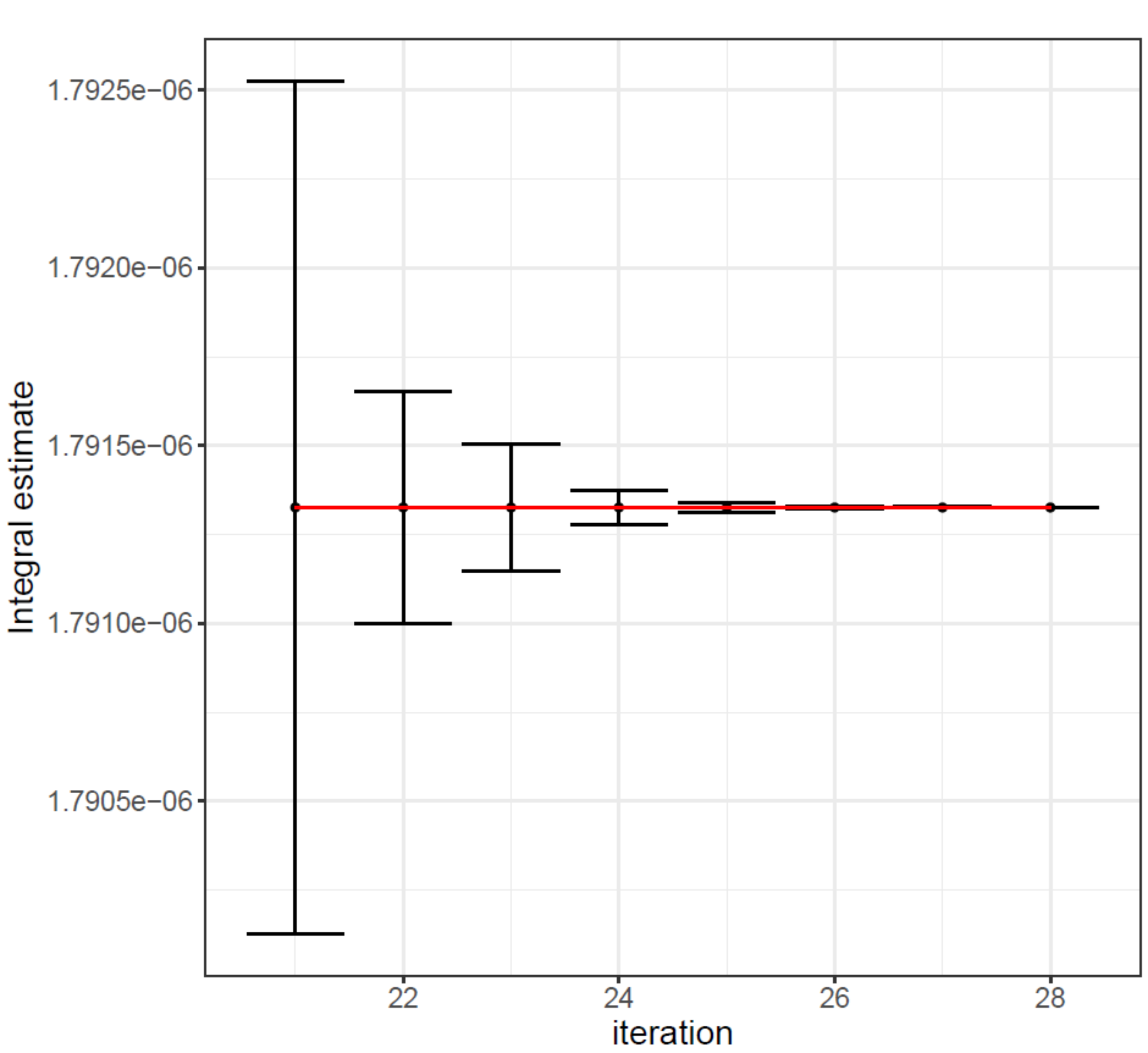
ACCURACY

- Estimates as measurement
- Consistency across iterations
- Overlapping error-bars
- 3-digits of precision
- Red line = true value



ACCURACY

- Estimates as measurement
- Consistency across iterations
- Overlapping error-bars
- 6-digits of precision
- Red line = true value
- More iterations



CONCLUSION

- New deterministic adaptive algorithm for highly parallel architectures
- No use of sequential algorithm
- Orders-of-magnitude speedup on challenging integrals
- Comparable performance with Two-Phase on low-precision
- 4-15 speedup over Two-Phase on medium/high precision
- Improved robustness and execution time
- Reliable error-estimation
- At least as reliable as Cuhre

FUTURE WORK

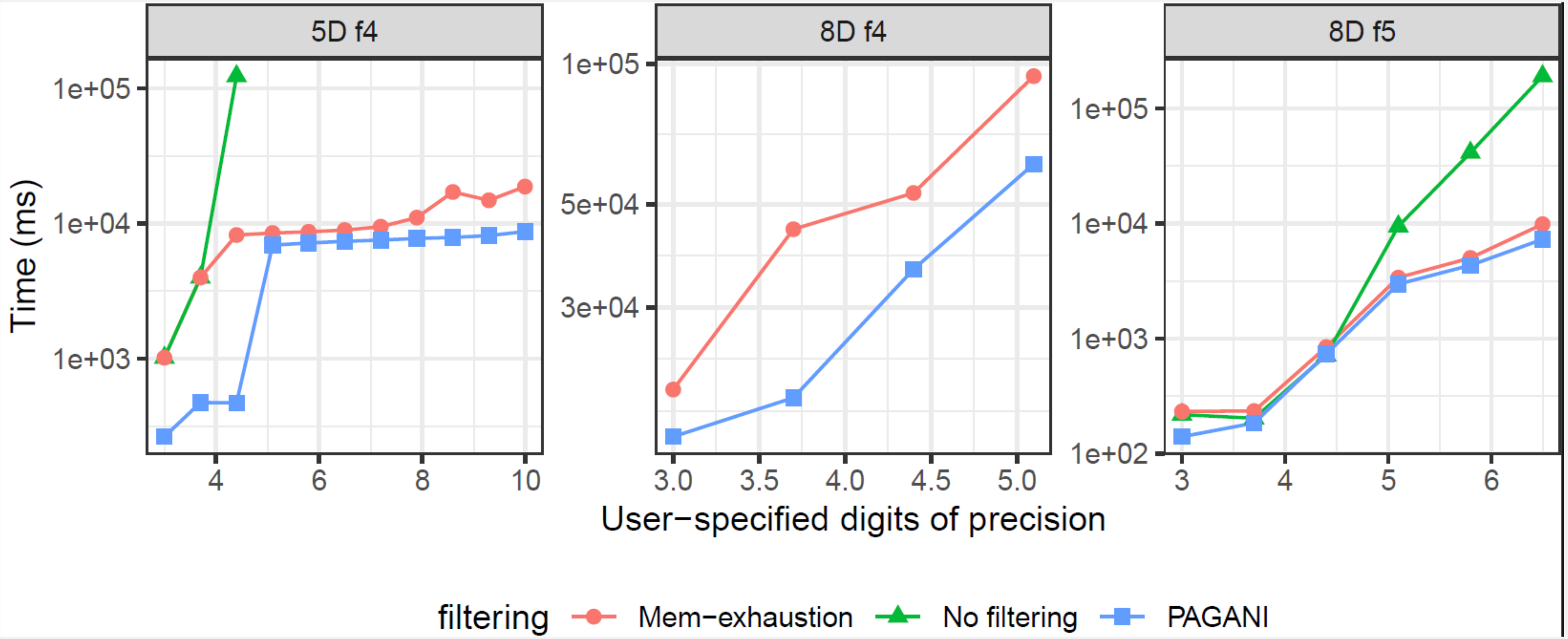
- Ampere architecture
- Multi-GPU
- Kokkos: 15% overhead in main computational kernels

QUESTIONS

IMPORTANT ALGORITHM CHARACTERISTICS

- All operations utilize parallelization
  - Function-evaluations within sub-region
  - Sub-region evaluations
- No use of sequential algorithm
- No persistence in region-processor mapping
- Global data-structure
- Global state through reduction
- Implicit synchronization

PERFORMANCE: HEURISTIC SEARCH FILTERING



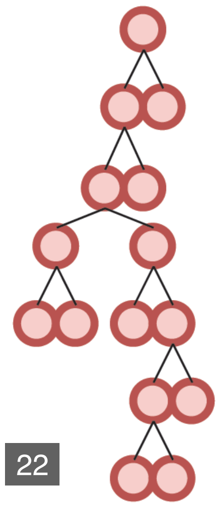
APPENDIX

BACKGROUND: CUHRE

- Maintains priority list based on highest error-estimate
- Sub-divides sub-region with highest error-estimate
- Sum integral estimate of all sub-regions to yield final integral estimate
- “Harder” integrals require more sub-regions
- Disjoint sub-region are independent

Algorithm 1 Sequential Adaptive Integration Algorithm

procedure ADAPTIVE NUMERICAL INTEGRATION(f, $b[n]$)
    $H \leftarrow b$
    while (termination condition is not satisfied) **do**
        Extract a non-empty subset of regions S from H
        for each region $R \in S$ **do**
            partition R into k regions along split-axis
            Let $R_1, R_2 \ldots R_k$ be these k regions
            for $i \leftarrow 1 \ldots k$ **do**
                $IntEst(R_i) \leftarrow$ Integral estimate for R_i
                $ErrEst(R_i) \leftarrow$ Error estimate for R_i
                $axis_i \leftarrow$ split-axis for R_i
                Insert R_i into H
    $GlobalIntEst \leftarrow \Sigma_{R_i \in H}\ IntEst(R_i)$
    $GlobalErrEst \leftarrow \Sigma_{R_i \in H}\ ErrEst(R_i)$
    return ($GlobalIntEst$, $GlobalErrEst$)



RELATIVE ERROR CLASSIFICATION

LEMMA 3.1. *Let m denote the number of subregions. Assume that $\forall i\ 1 \le i \le m$, $e_i \ge 0$ and v_i's have the same sign. Let $e = \sum_{i=1}^{m} e_i$ denote the cumulative error and $v = \sum_{i=1}^{m} v_i$ denote the cumulative integral estimate. Suppose $\forall i\ 1 \le i \le m$, $e_i \le |v_i| \cdot \tau_{rel}$. Then $e \le |v| \cdot \tau_{rel}$.*

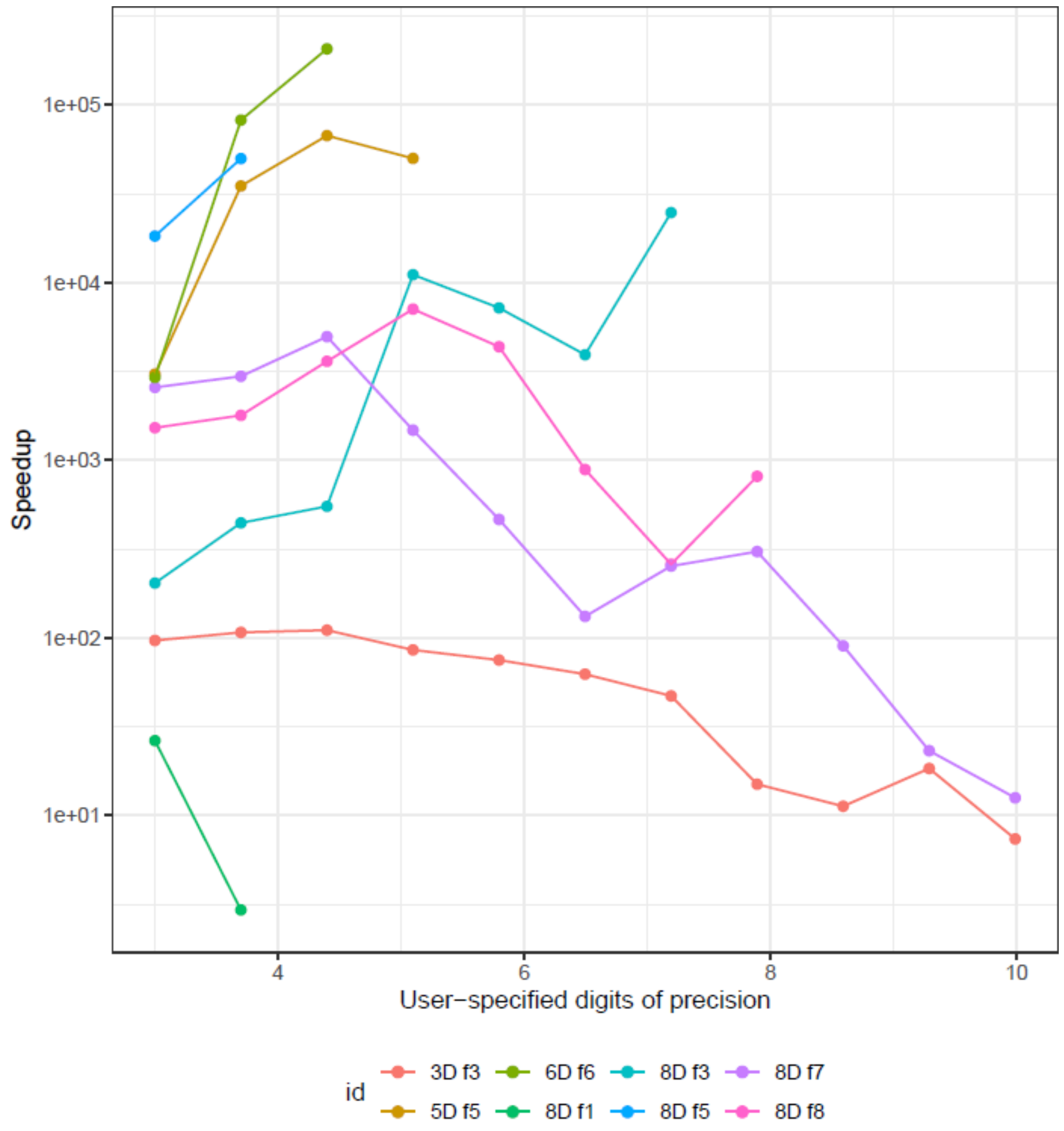
PROOF. $e = \sum_{i=1}^{m} e_i \le \sum_{i=1}^{m} |v_i| \cdot \tau_{rel} \le |\sum_{i=1}^{m} v_i| \cdot \tau_{rel} = |v| \cdot \tau_{rel}$. □

PURE ERROR ESTIMATE

- Error coefficients
- If $coeff[0] * rule[1] \leq rule[2] \;\; coeff[0] * rule[2] \leq sum[3]$
  - Return $coeff[1] * rule[1]$
  - Otherwise return $coeff[2] * \max(\max(rule[1],\ rule[2]),\ rule[3])$

TWO-LEVEL ERROR ESTIMATE

- Pure error-estimate
- Integral estimate
- Sibling estimate
- Parent estimate
- Diff: $\left| .25(\text{siblRes} + \text{selfRes} - \text{parRes}) \right|$
- C = 1 + 2diff/(selfErr+siblErr)
- selfErr = c*selfErr
- selfErr == selfErr + diff



SPEEDUP OVER QUASI-MONTE CARLO

PERFORMANCE BREAKDOWN

- Sub-region Evaluation: more than **90%** of execution time
  - 40% peak-performance for double-precision
  - Compute-bound
  - Limited number of memory accesses (initial read, final write)
- Post-processing, classification, filtering, sub-division
  - Thrust library
- All operations on GPU

Algorithm 2 PAGANI Algorithm

procedure PAGANI(f, n, $b[n]$, τ_{rel}, τ_{abs})
  $R_0 \leftarrow b$
  $s \leftarrow d^n$ ▹ region list size
  $H \leftarrow$ UNIFORM-SPLIT(R_0, d)
  $A[1:s] \leftarrow 1$
  $V[1:s], E[1:L], K[1:s] \leftarrow 0$
  $V_p[1:s], E_p[1:s] \leftarrow 0$
  $v, e, v_f, e_f \leftarrow 0$ ▹ cumulative/finished estimates
  for $it : it_{max}$ **do**
    $V, E, K \leftarrow$ EVALUATE(H)
    $E < -$ TWO-LEVEL-ERROR(V, E, V_p, E_p)
    $A < -$ REL-ERR-CLASSIFY(V, E, A)
    $v \leftarrow$ SUM(V)
    $e \leftarrow$ SUM(E)
    if $\frac{e+e_f}{|v+v_f|} \leq \tau_{rel}$ or $e + e_f \leq \tau_{abs}$ **then**
      `return` $v + v_f, e + e_f$
    $A \leftarrow$ THRESHOLD-CLASSIFY ($A, E, v + v_f, e + e_f, v, e, s$)
    $v_f \leftarrow v-$ SUM($V \cdot A$) $+v_f$
    $e_f \leftarrow e-$ SUM($E \cdot A$) $+e_f$
    $H, V, E, L \leftarrow$ FILTER(H, V, E, A)
    $V_p \leftarrow V, E_p \leftarrow E$ ▹ update all parents
    $H \leftarrow$ SPLIT(H, K)
    $s \leftarrow 2s$

Algorithm 3 Threshold Classification Algorithm

procedure THRESHOLD-CLASSIFY($A, E, v_{tot}, e_{tot}, e_{it}, s_{it}$)
    $P_{max} \leftarrow .25$ ▹ target percentage of error budget
    $e_{\bar{a}} \leftarrow 0$ ▹ error-estimate contribution from inactive regions
    $s_{\bar{a}} \leftarrow 0$ ▹ # inactive regions
    $min, max \leftarrow MinMax(E)$
    $e_b \leftarrow e_{tot} - v_{tot} \cdot \tau_{rel}$ ▹ error budget
    $t \leftarrow \frac{e_{it}}{s_{it}}$ ▹ set initial threshold as avg. error-estimate
    repeat
        $A \leftarrow$ APPLY-THRESHOLD(E, t)
        $s_{\bar{a}} \leftarrow n-$ SUM(A)
        if $s_{\bar{a}} > .5 \cdot s_{it}$ **then** ▹ memory requirement
            $e_{\bar{a}} \leftarrow e_{it}-$ SUM($A \cdot E$)
            if $e_{\bar{a}} \leq P_{max} \cdot e_b$ **then** ▹ accuracy requirement
                return A
        UPDATE-THRESHOLD($t, s_{\bar{a}}, e_{\bar{a}}, P_{max}, min, max$)
    until $\frac{s_{\bar{a}}}{s} > .5$ and $e_{\bar{a}} \leq P_{max} \cdot e_b$
    return A ▹ unsuccessful filtering

HEURISTIC THRESHOLD SEARCH

- Classify many regions with “small” contributions
- How small is small enough?
- Try a different values and observe effect before committing to filtering
- Heuristic filtering: average error-estimate as initial threshold
- Try current threshold
- If % of finished regions is too small, move towards max
- Else move towards min
- Try again