Order-Invariant Real Number Summation: Circumventing Accuracy Loss for Multimillion Summands on Multiple Parallel Architectures

Patrick E. Small, Rajiv K. Kalia, Aiichiro Nakano, Priya Vashishta

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Biological Sciences
University of Southern California

Email: (patrices, rkalia, anakano, priyav)@usc.edu

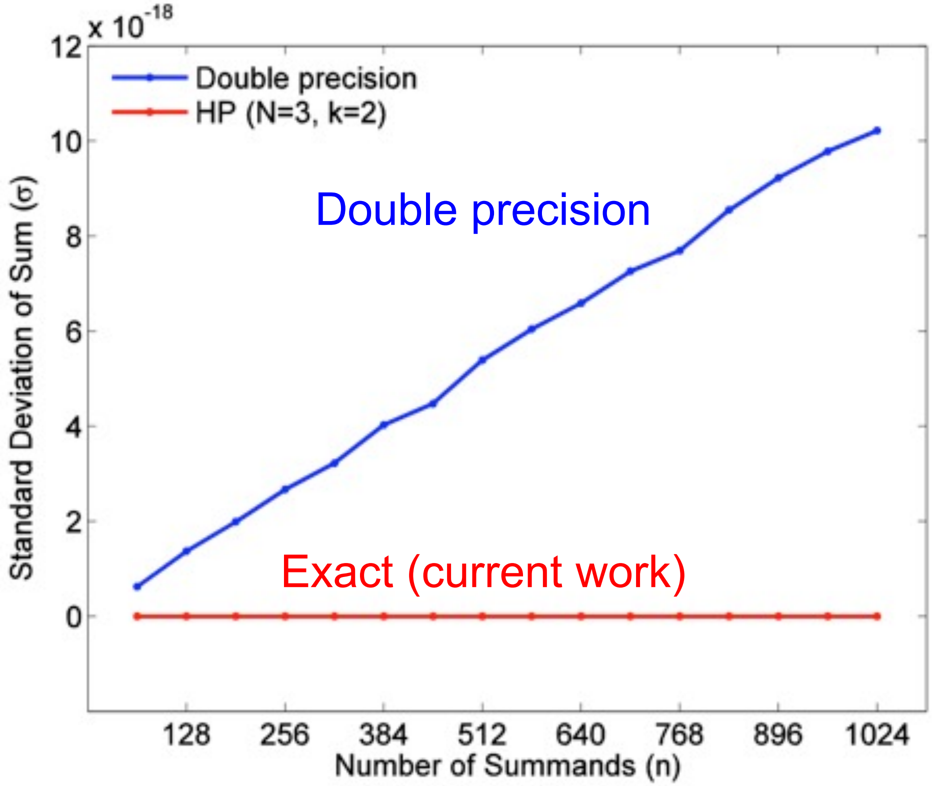
Proc. IEEE International Parallel & Distributed Processing Symposium, IPDPS, p. 152 ('16)
https://aiichironakano.github.io/cs596/Small-OrderInvariantSum-IPDPS16.pdf

Optimize bits!

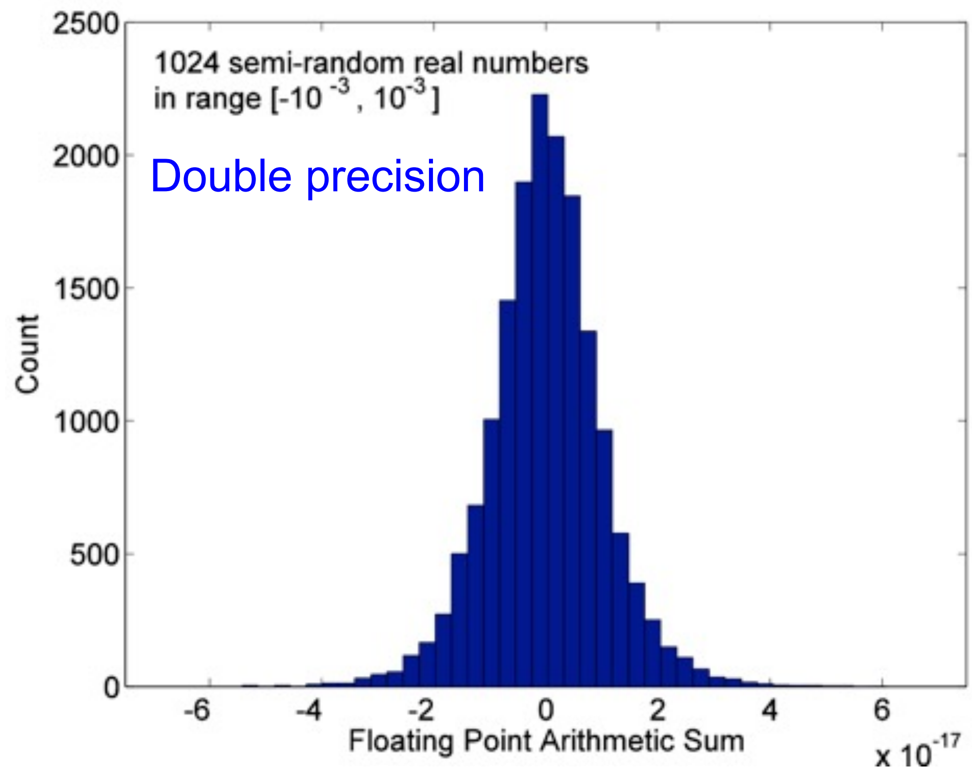
Reproducibility Challenge

- **Rounding (truncation) error makes floating-point addition non-associative**

$$(a + b) + c \neq a + (b + c)!$$



Standard deviation of sum with random summation orders



Distribution of sum with random summation orders

- **Finding: Sum becomes a random walk across the space of possible rounding error**

High-Precision (HP) Method

- **Propose an extension of the order-invariant, higher-precision intermediate-sum method by Hallberg & Adcroft** *Par. Comput*. **40**, 140 ('14)
- **The proposed variation represents a real number *r* using a set of *N* 64-bit unsigned integers, a_i (i = 0, *N*-1)**

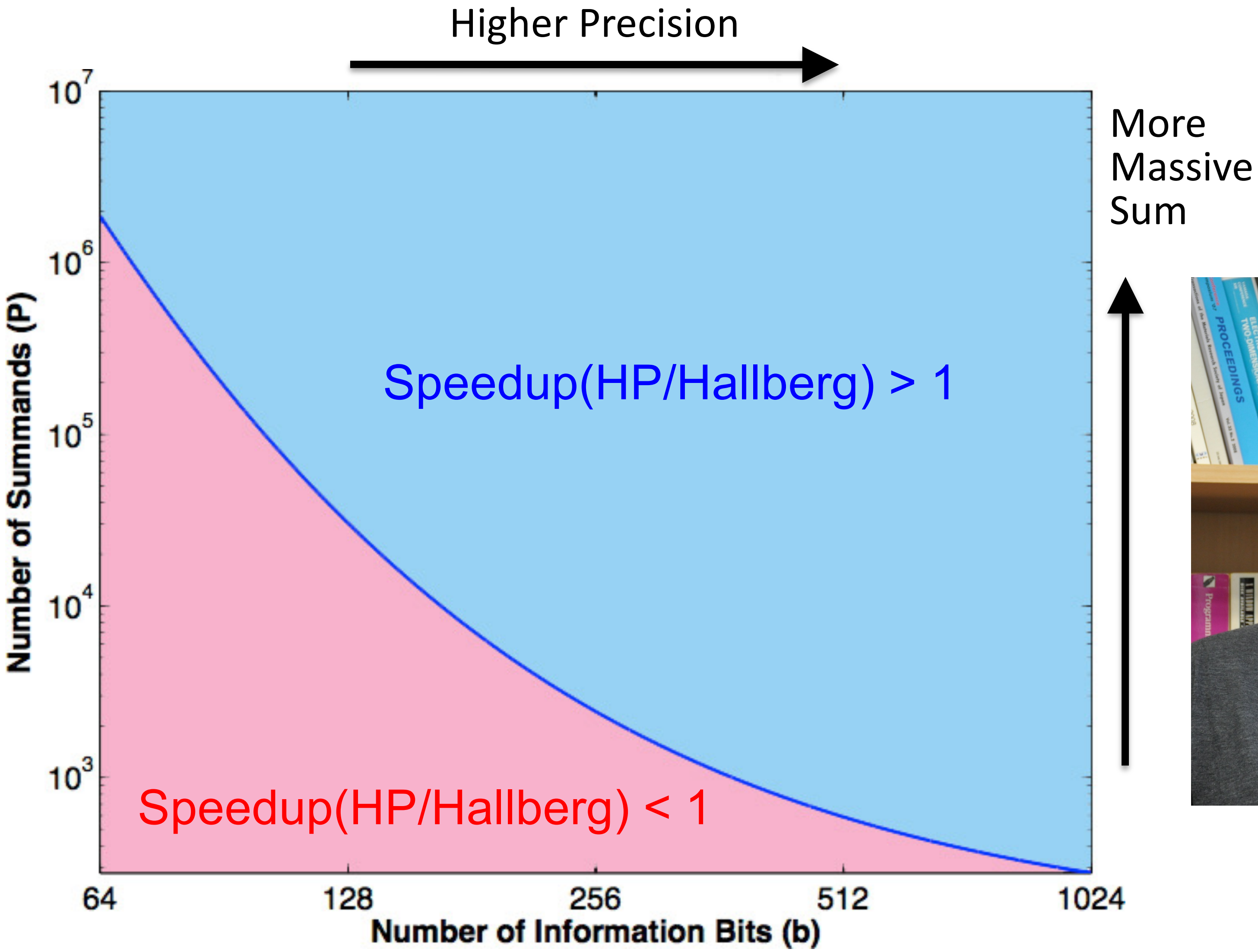
$$r = \sum_{i=0}^{N-1} a_i 2^{64(N-k-i-1)}$$
$$= \overbrace{a_0 2^{64(N-k-1)} + \cdots + a_{N-k-1}}^{N-k} + \cdots + \overbrace{a_{N-k} 2^{-64} + \cdots + a_{N-1} 2^{-64k}}^{k}$$

- ***k* is the number of 64-bit unsigned integers assigned to represent the fractional portion of *r* (0 ≤ *k* ≤ *N*), whereas *N*-*k* integers represent the whole-number component**
- **Negative number is represented by two's complement in integer representation, using only 1 bit**

If you are the first to find the problem, the simplest solution suffices to prove the concept

Performance Projection

- **HP sum is faster than Hallberg sum for higher precision & larger numbers of summands**



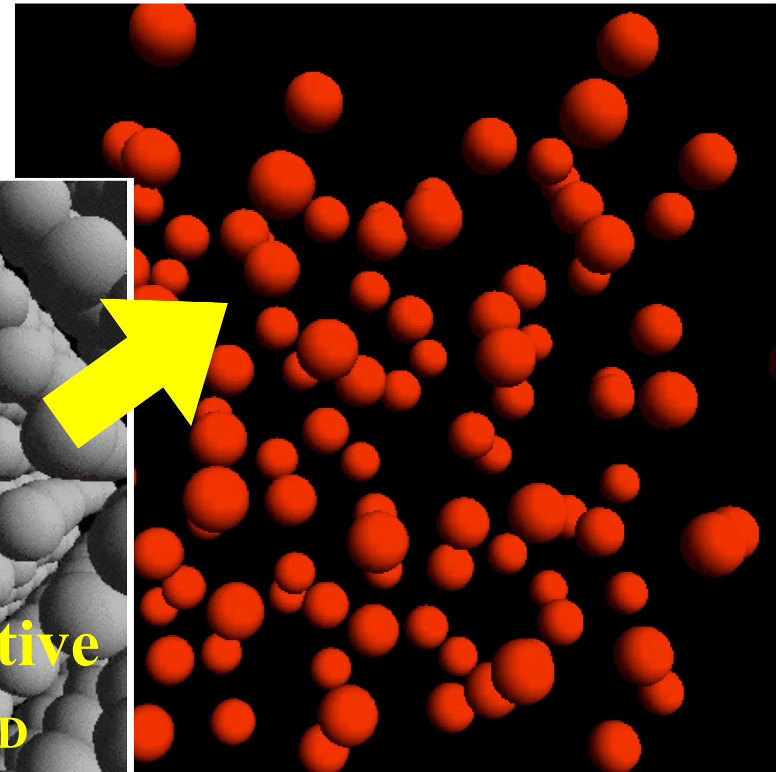
P. E. Small *et al.*, *Proc. IEEE IPDPS*, p. 152 ('16)

Detailed IPDPS 2016 Presentation

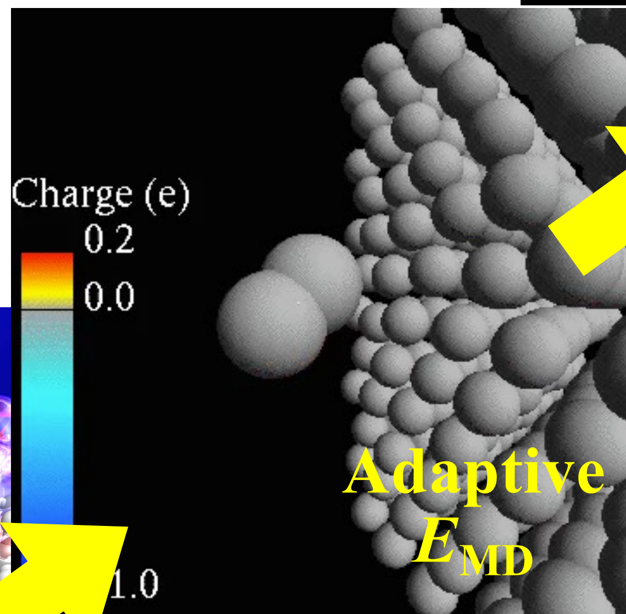
RIP, Patrick

Hierarchy of Atomistic Simulation Methods

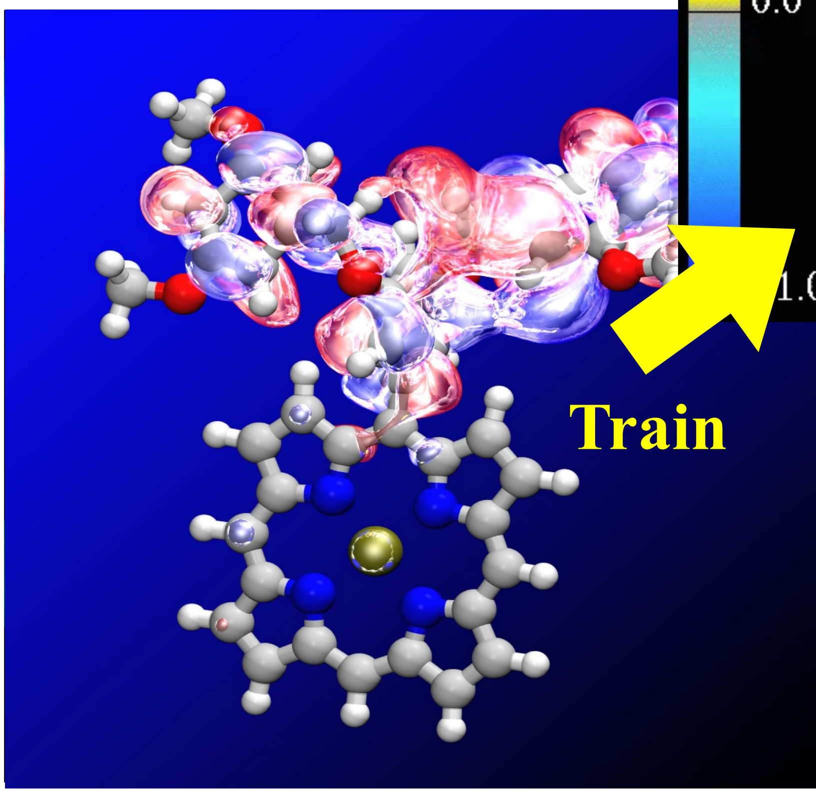
Molecular Dynamics (MD)



Reactive MD (RMD)



Nonadiabatic quantum MD (NAQMD)

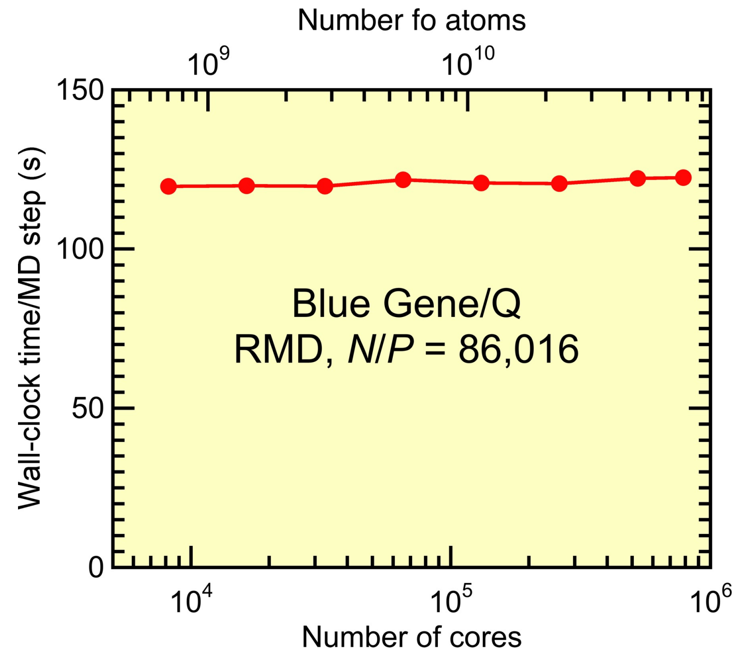


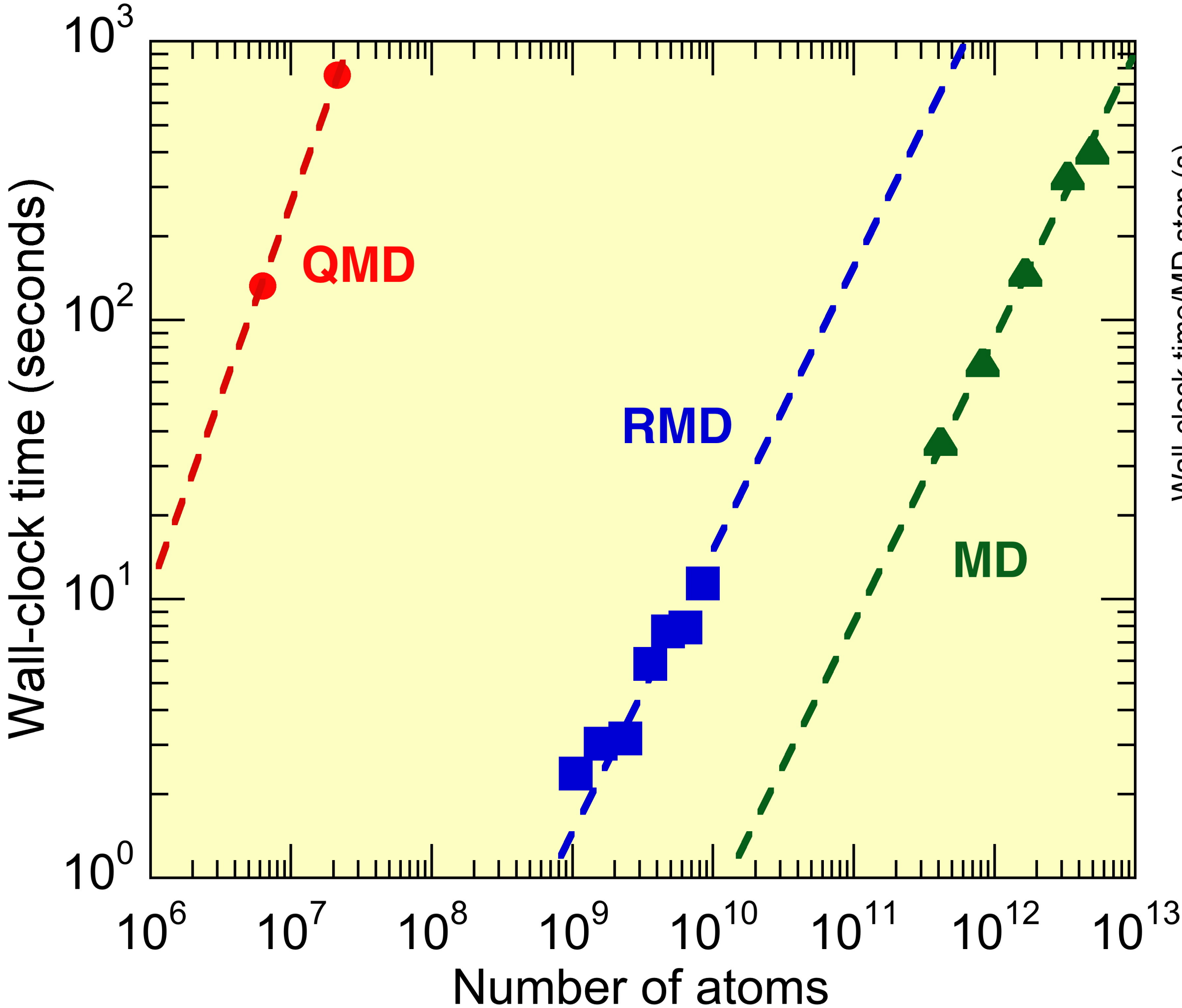
First principles-based reactive force-fields

- Reactive bond order $\{BO_{ij}\}$
  → Bond breakage & formation
- Charge equilibration (QEq) $\{q_i\}$
  → Charge transfer

Tersoff, Brenner, Sinnott *et al.*; Streitz & Mintmire *et al.*; van Duin & Goddard (ReaxFF)

Scalable Simulation Algorithm Suite





QMD (quantum molecular dynamics): DC-DFT

RMD (reactive molecular dynamics): F-ReaxFF

MD (molecular dynamics): MRMD

- 4.9 trillion-atom space-time multiresolution MD (MRMD) of SiO_2
- 67.6 billion-atom fast reactive force-field (F-ReaxFF) RMD of RDX
- 39.8 trillion grid points (50.3 million-atom) DC-DFT QMD of SiC

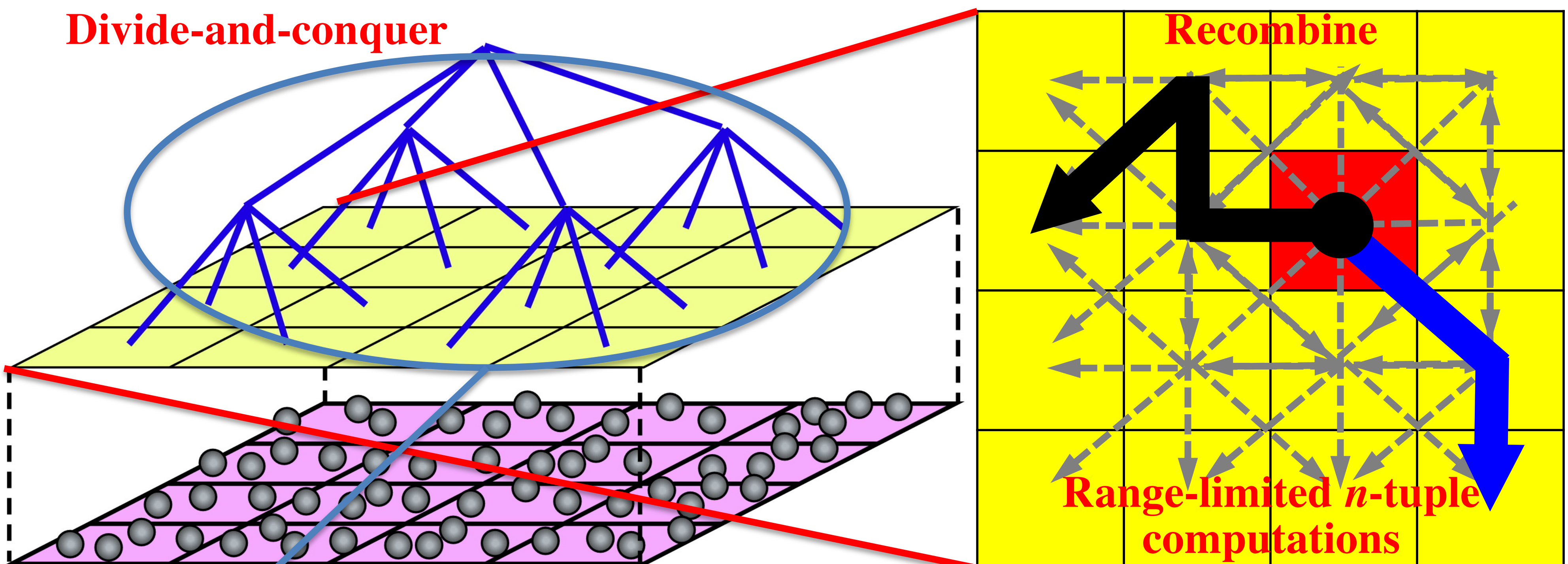
parallel efficiency 0.984 on 786,432 Blue Gene/Q cores

Exascale Computing Challenge

1. Scalability beyond million-way parallelism

J. Chem. Phys. **140**, 18A529 ('14)
IEEE/ACM SC14
IEEE Computer **48(11)**, 33 ('15)

Divide-conquer-recombine (DCR) algorithmic framework
Metascalable ("design once, scale on future architectures")

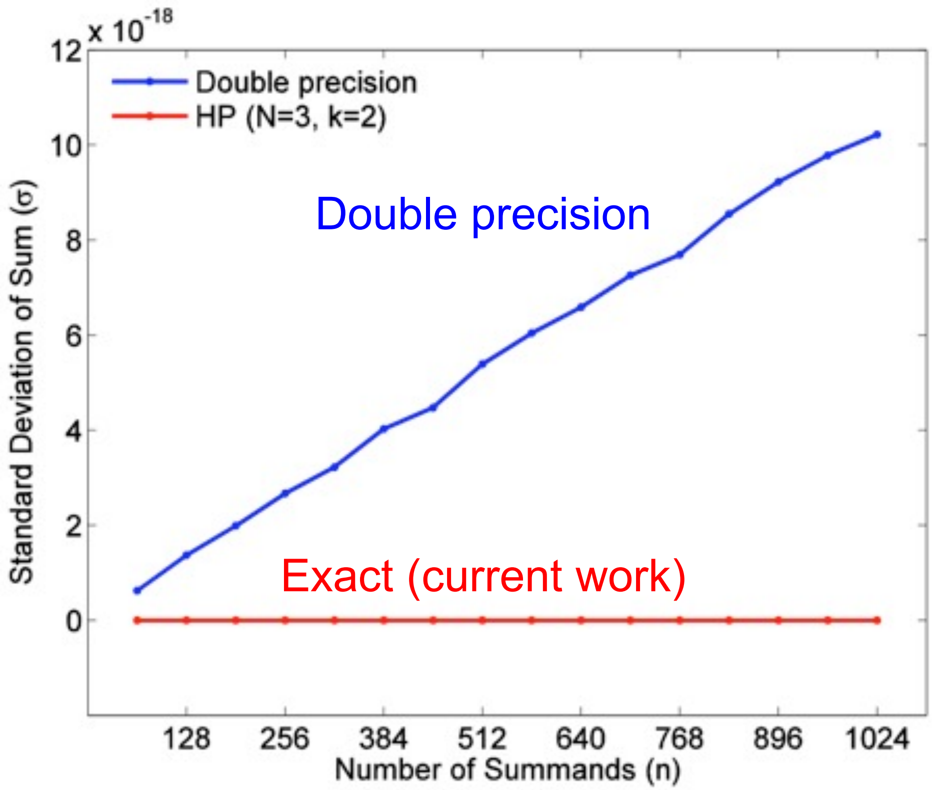


ACM/IEEE SC13

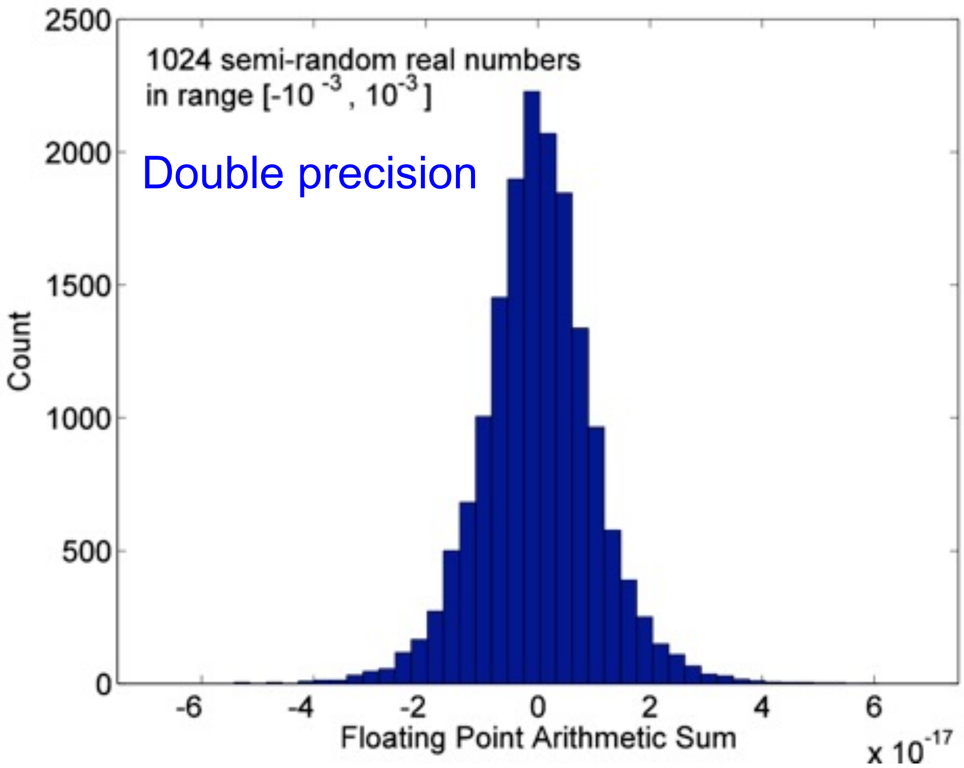
2. Reproducibility of real-number summation for multimillion summands & beyond in the global sum; double-precision arithmetic began to produce different results on different high-end architectures

Reproducibility Challenge

- **Rounding (truncation) error makes floating-point addition non-associative**



Standard deviation of sum with random summation orders



Distribution of sum with random summation orders

- **Sum becomes a random walk across the space of possible rounding error**

Related Works

- **General-purpose arbitrary precision arithmetic**
  [GNU-MPL ’12]
  → **Extensive computation & memory usage**
- **Error-compensation methods**
  > **Error-free transformation for tracking residuals**
  > [Priest, ’91, Higham ’93, Rump ’09, Demmel ’13]
  > → **Complex implementation**

  > **Summation reordering for minimizing error**
  > [Hel ’01]
  > → **Prohibitive at large scales**
- **Hardware solutions**
  [Gustafson ’15]
  → **Not available yet**
- **Higher-precision intermediate sums**
  [He ’01, Hallberg ’14]
  → **Simple implementation, low overhead**

Contributions

- **Propose an extension of the order-invariant, higher-precision intermediate-sum method by Hallberg & Adcroft** [*Par. Comput*. **40**, 140 ('14)]**:**
  - **(1) Improves performance* for large (> 10^6) number of summands**
  - **(2) Eliminates the aliasing problem of the original method**
- **The new method outperforms the previous state-of-the-art for large problems involving million+ summands on broad systems (MPI, OpenMP, CUDA/GPU, Xeon Phi)**

*Performance is defined as the computational speed

Hallberg Order-Invariant Sum

- **Integer representation with higher accuracy: Represent a real number *r* using a set of *N* 64-bit signed integers, a_i (*i* = 0, *N*-1); *M* (< 63) is a positive integer**

$$r = \sum_{i=0}^{N-1} a_i 2^{\left(i-\frac{N}{2}\right)M} = 2^{-NM/2}(a_0 + a_1 2^M + a_2 2^{2M} + \cdots)$$

- **Order-invariant parallel sum: Two real numbers are added by summing *N* pairs of corresponding integers concurrently**

- **Carry out (potential sequential dependence): When any of the integer additions exceeds 2^M, carry out must be added to the next integer in the set**

- **Carry-overhead reduction: Carry operations are avoided up to $P = 2^{63-M}-1$ summands to expose high parallelism**

R. Hallberg & A. Adcroft, *Par. Comput*. **40**, 140 ('14)

Drawback of Hallberg Sum

- **Overhead:** Not all integer bits serve to provide real-number precision; 63-M bits per integer are dedicated to book-keeping
- **Aliasing:** Multiple integer representations could represent the same real number
- Normalization & sum overheads to convert the integer representation back to real

High-Precision (HP) Method

- **The proposed variation of Hallberg method represents a real number *r* using a set of *N* 64-bit unsigned integers, a_i (*i* = 0, *N*-1)**

$$r = \sum_{i=0}^{N-1} a_i 2^{64(N-k-i-1)}$$

$$= \underbrace{a_0 2^{64(N-k-1)} + \cdots + a_{N-k-1}}_{N-k} + \underbrace{a_{N-k} 2^{-64} + \cdots + a_{N-1} 2^{-64k}}_{k}$$

- ***k* is the number of 64-bit unsigned integers assigned to represent the fractional portion of *r* (0 ≤ *k* ≤ *N*), whereas *N*-*k* integers represent the whole-number component**
- **Negative number is represented by two's complement in integer representation, using only 1 bit**

HP Algorithm (1): Conversion

- **Simple procedure: A single pass converts a double-precision number r to HP integers a_i & translates them to two's complement**

$$r = \sum_{i=0}^{N-1} a_i 2^{64(N-k-i-1)}$$

```
dtmp = fabs(r)*2^(64*(N-k-1));
isneg = (r < 0.0);
for (i=0; i<N-1; i++) {
    itmp = (uint64_t)dtmp;
    dtmp = (dtmp – (double)itmp)*2^64;
    a[i] = (isneg) ? ~itmp + (dtmp<=0.0) : itmp;
}
a[N-1] = (isneg) ? ~(uint64_t)dtmp + 1 : (uint64_t)dtmp;
```

- **Inverse of this algorithm converts HP number back to double-precision**

HP Algorithm (2): Addition

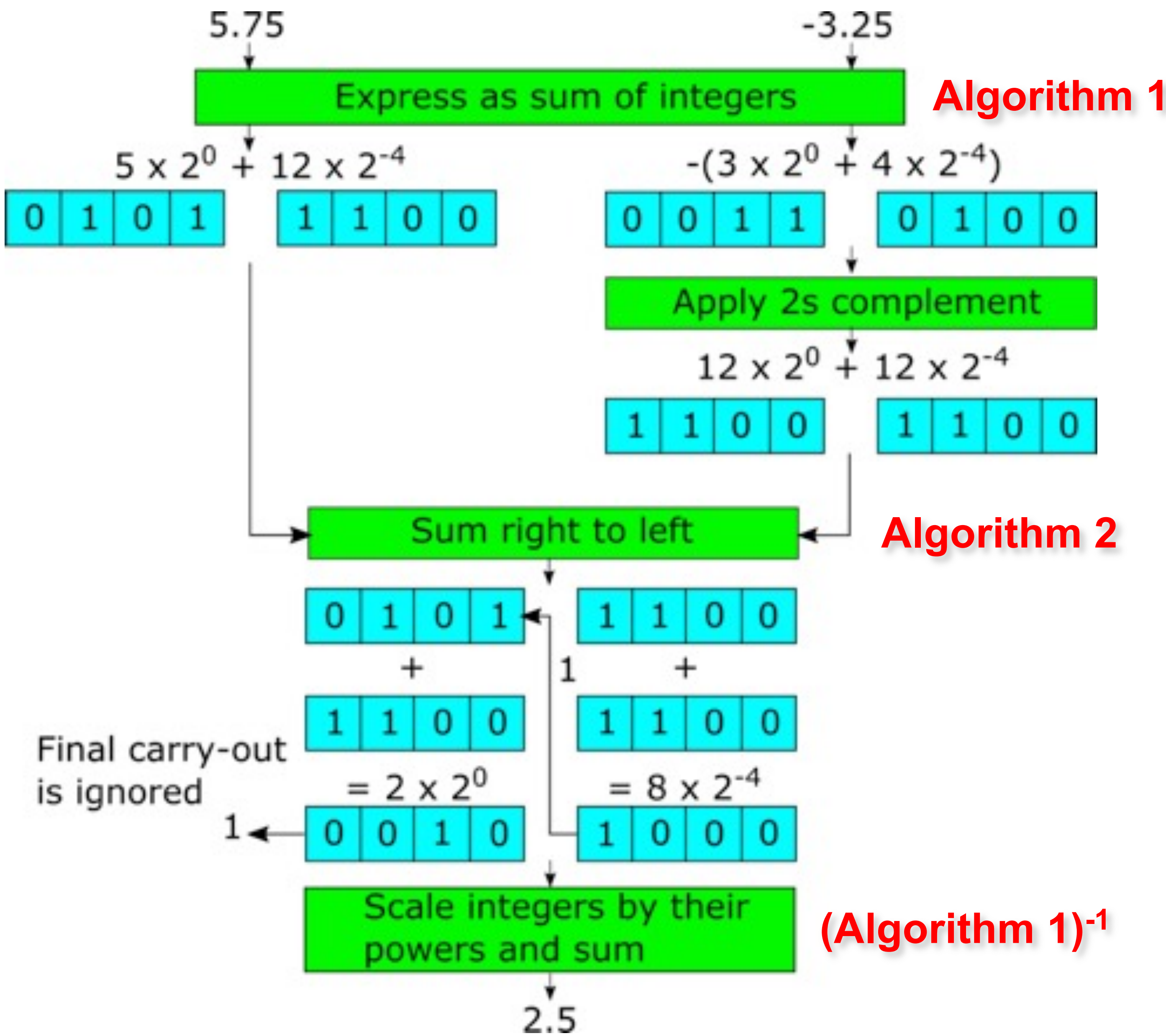
- **Addition of two HP numbers, $a \leftarrow a + b$**

$$\begin{cases} r_1 = \sum_{i=0}^{N-1} a_i 2^{64(N-k-i-1)} \\ r_2 = \sum_{i=0}^{N-1} b_i 2^{64(N-k-i-1)} \end{cases}$$

```
a[N-1] = a[N-1]+b[N-1];
co = (a[N-1]<b[N-1]);
for (i=N-2; i>=1; i--) {
    a[i] = a[i]+b[i]+co;
    co = (a[i]==b[i]) ? co : (a[i]<b[i]);
}
a[0] = a[0]+b[0]+co;
```

- **Overflow of the sum is detected by comparing the signs of the summands with that of the sum**

HP Sum: Example



Representation Power

- **Maximum range & smallest representable HP number**

$$r_{\text{HP}} = \sum_{i=0}^{N-1} a_i 2^{64(\boxed{N}-\boxed{k}-i-1)}$$

N	k	Bits	Maximum range	Smallest number
2	1	128	$\pm 9.223372\times 10^{18}$	5.421011×10^{-20}
3	2	192	$\pm 9.223372\times 10^{18}$	2.938736×10^{-39}
6	3	256	$\pm 3.138551\times 10^{57}$	1.593092×10^{-58}
8	4	512	$\pm 5.789604\times 10^{76}$	8.636169×10^{-78}

- **Equivalency with Hallberg representation power**

$$r_{\text{Hallberg}} = \sum_{i=0}^{N-1} a_i 2^{(i-\boxed{N}/2)\boxed{M}}$$

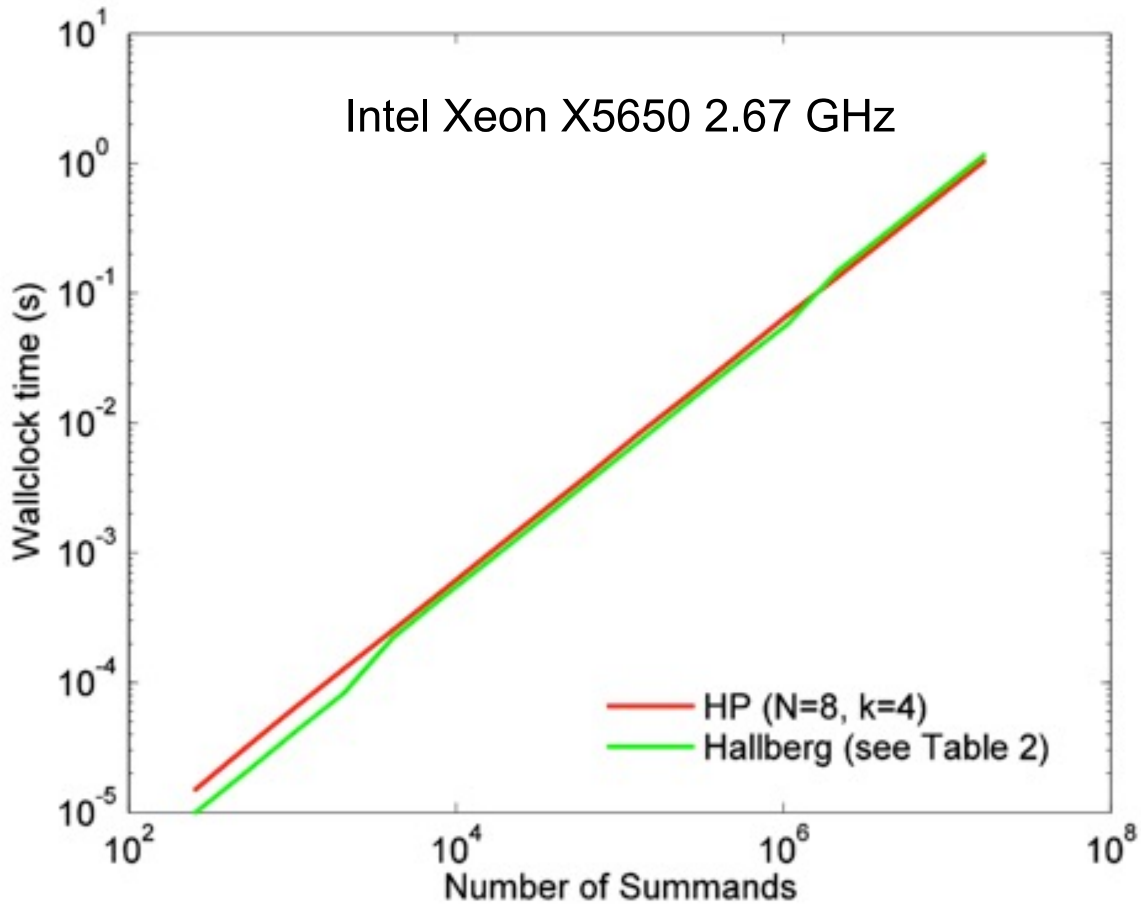
N	M	Precision bits	Max summands
10	52	520	2048
12	43	516	1 M
14	37	518	64 M

HP Method: Properties

- **Invariance of sum with respect to both summation order & architecture is guaranteed with appropriate setting of *N* & *k* to provide sufficient accuracy**
- **Overflow & underflow can be readily detected at runtime at double-precision (DP)-to-HP conversion, HP-sum & HP-to-DP conversion steps**
- **Atomicity of addition (which is essential for multithreading) is guaranteed using only the widely available compare-&-swap (CAS) synchronization primitive**

Performance

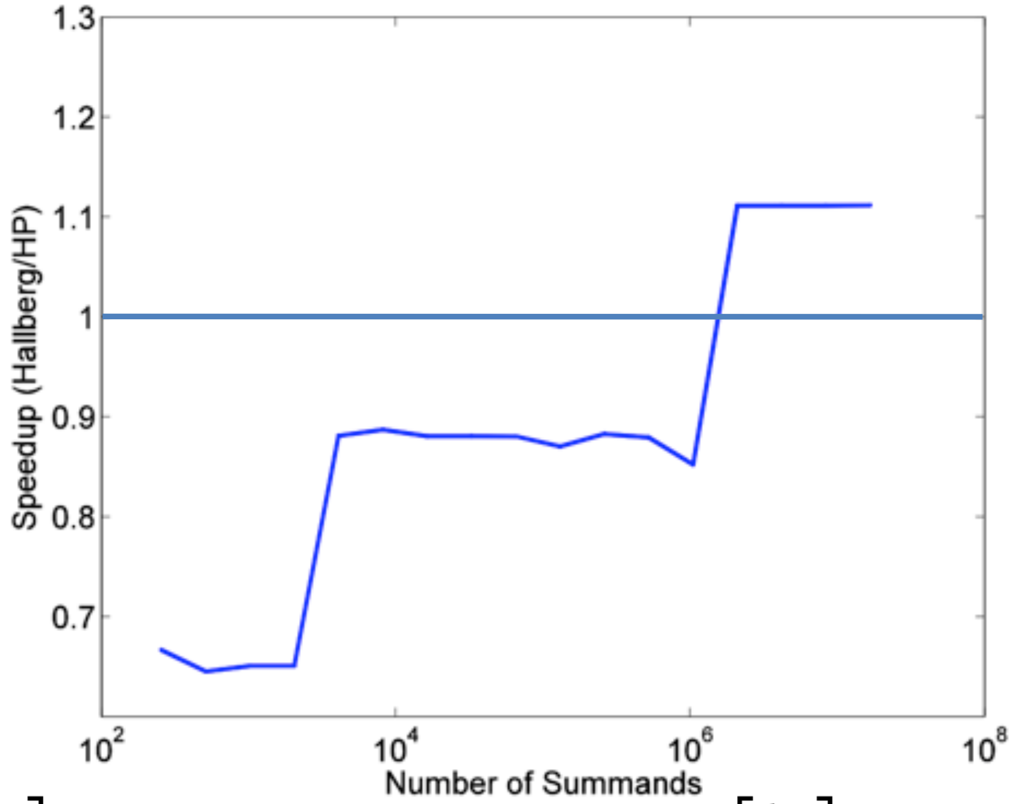
- Computing time of real-number sum using the current (HP) & Hallberg methods as a function of # of summands



- HP sum is faster than Hallberg sum over million summands

Performance Analysis

- **Speedup of HP sum over Hallberg sum as a function of the number of summands**

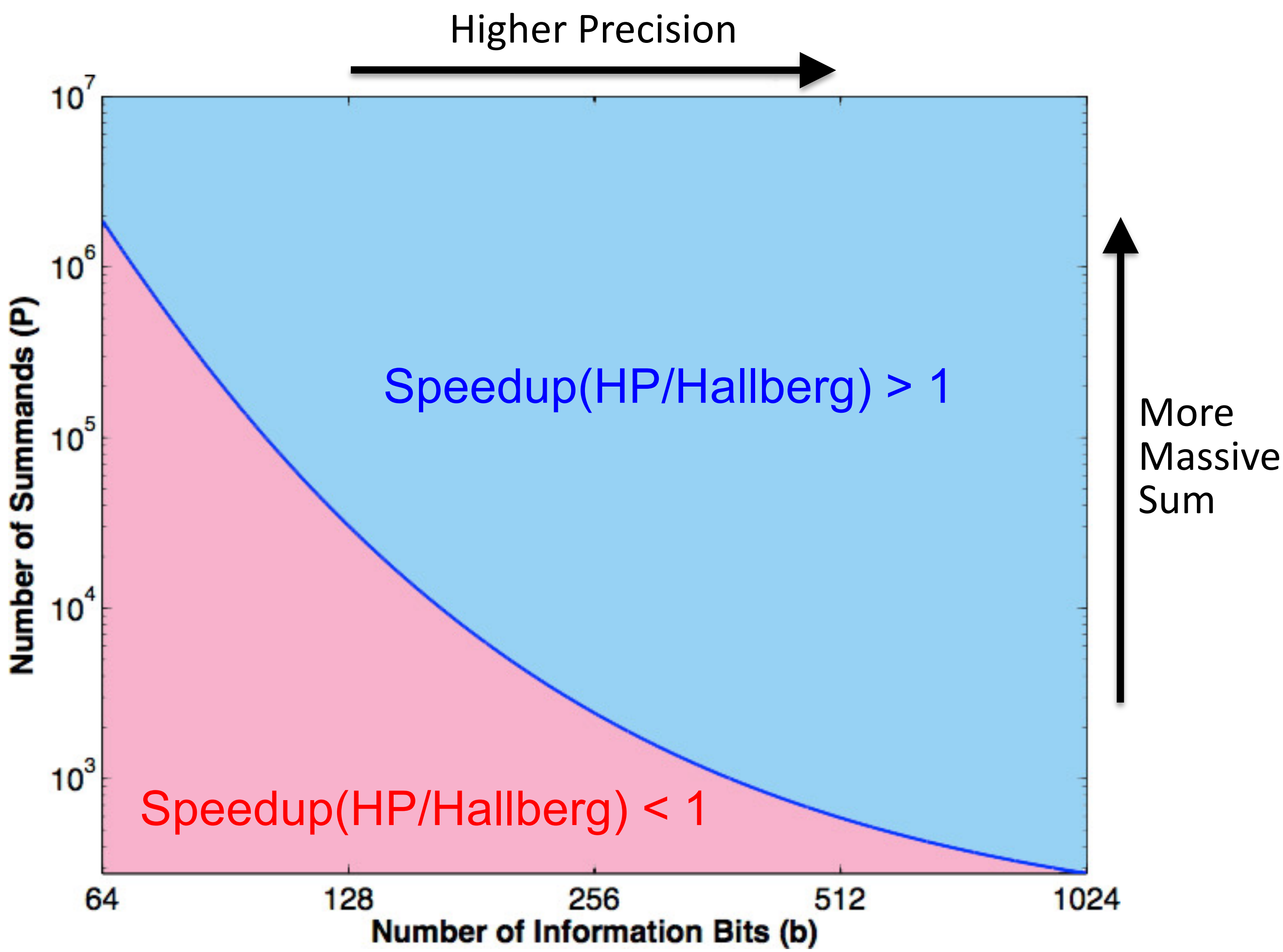


$$T_{\text{HP}} = c_{\text{HP}} \left\lceil \frac{b+1}{64} \right\rceil \qquad T_{\text{Hallberg}} = c_{\text{Hallberg}} \left\lceil \frac{b}{M} \right\rceil$$ b: Precision bit count

$$\text{Speedup} \sim \frac{c_{\text{Hallberg}}}{c_{\text{HP}}} \frac{64b}{M(b+65)} \geq \frac{32 c_{\text{Hallberg}}}{c_{\text{HP}}} \frac{1}{M} \qquad 2^{63\text{-}M} \propto \#\text{summands}$$

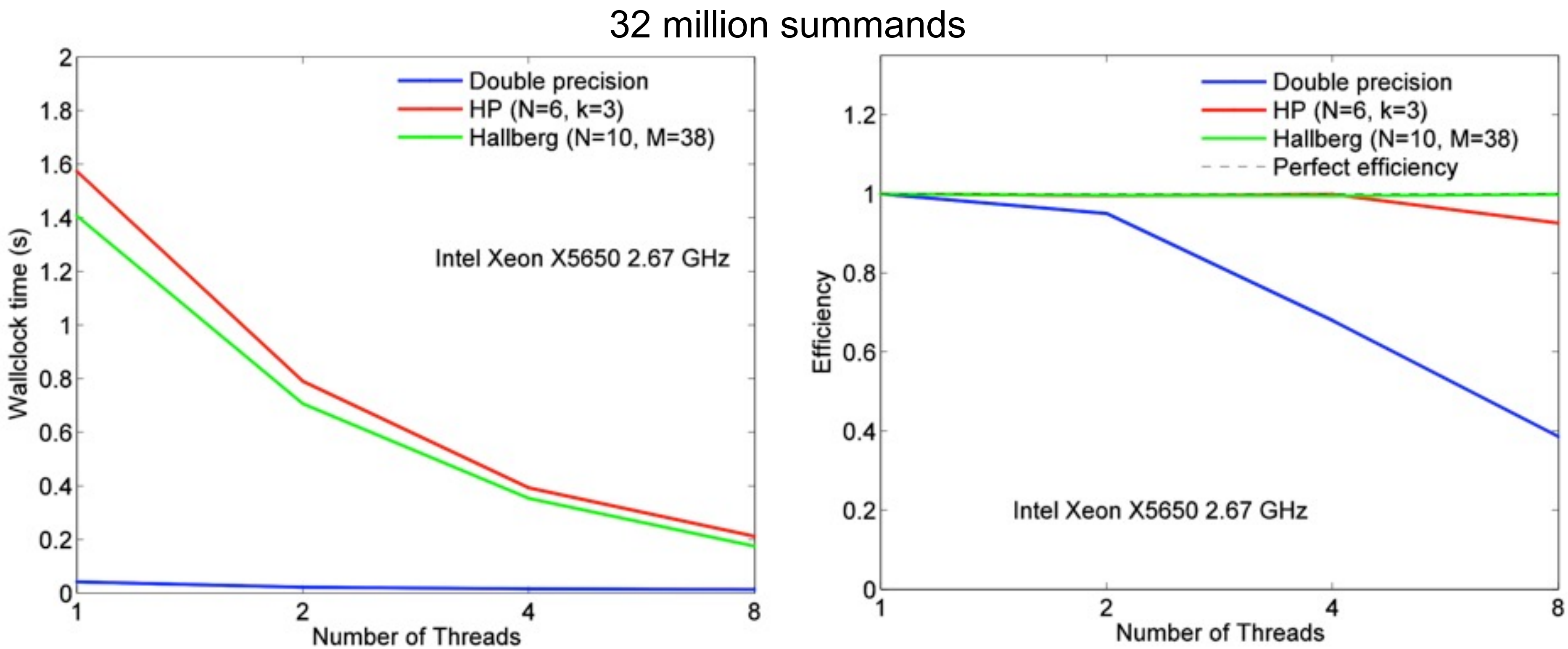
Performance Projection

- **HP sum is faster than Hallberg sum for larger numbers of summands & higher precision**



Parallel Efficiency with OpenMP

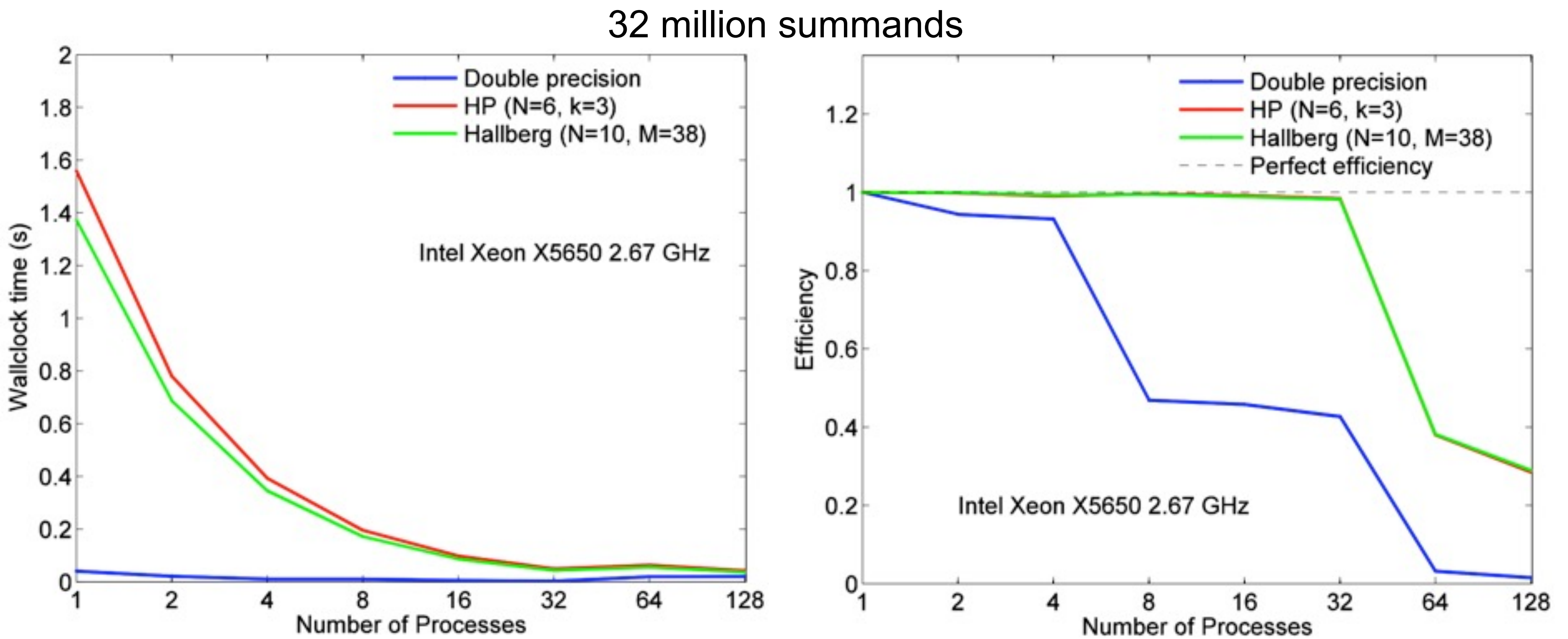
- **Runtime & strong-scaling parallel efficiency of HP, Hallberg & (order-sensitive) double-precision sums as a function of the number of OpenMP threads on Xeon**



- **Higher parallel efficiency of HP & Hallberg sums over double-precision sum**

Parallel Efficiency with MPI

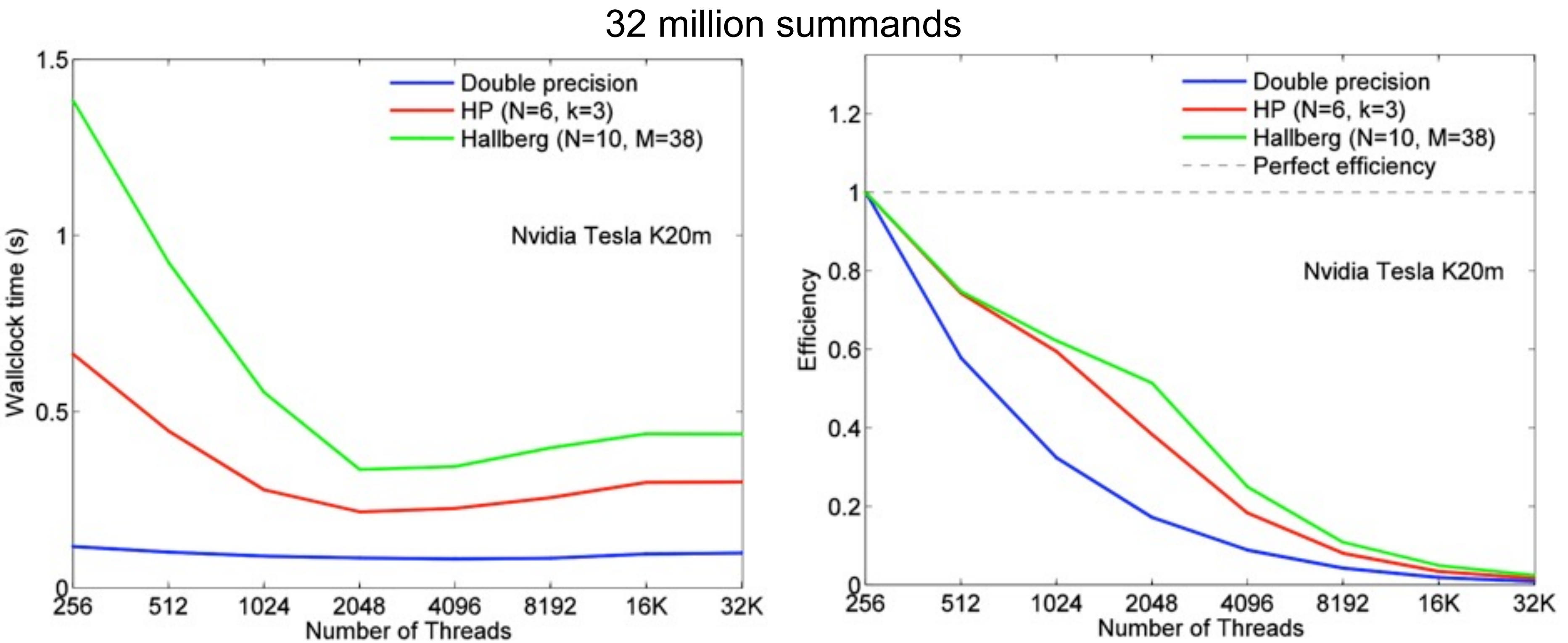
- **Runtime & strong-scaling parallel efficiency of HP, Hallberg & (order-sensitive) double-precision sums as a function of the number of MPI processes on Xeon**



- **Higher parallel efficiency of HP & Hallberg sums over double-precision sum**

Parallel Efficiency on GPGPU

- **Runtime & strong-scaling parallel efficiency of HP, Hallberg & (order-sensitive) double-precision sums as a function of the number of CUDA threads on general-purpose graphics processing unit (GPGPU)**

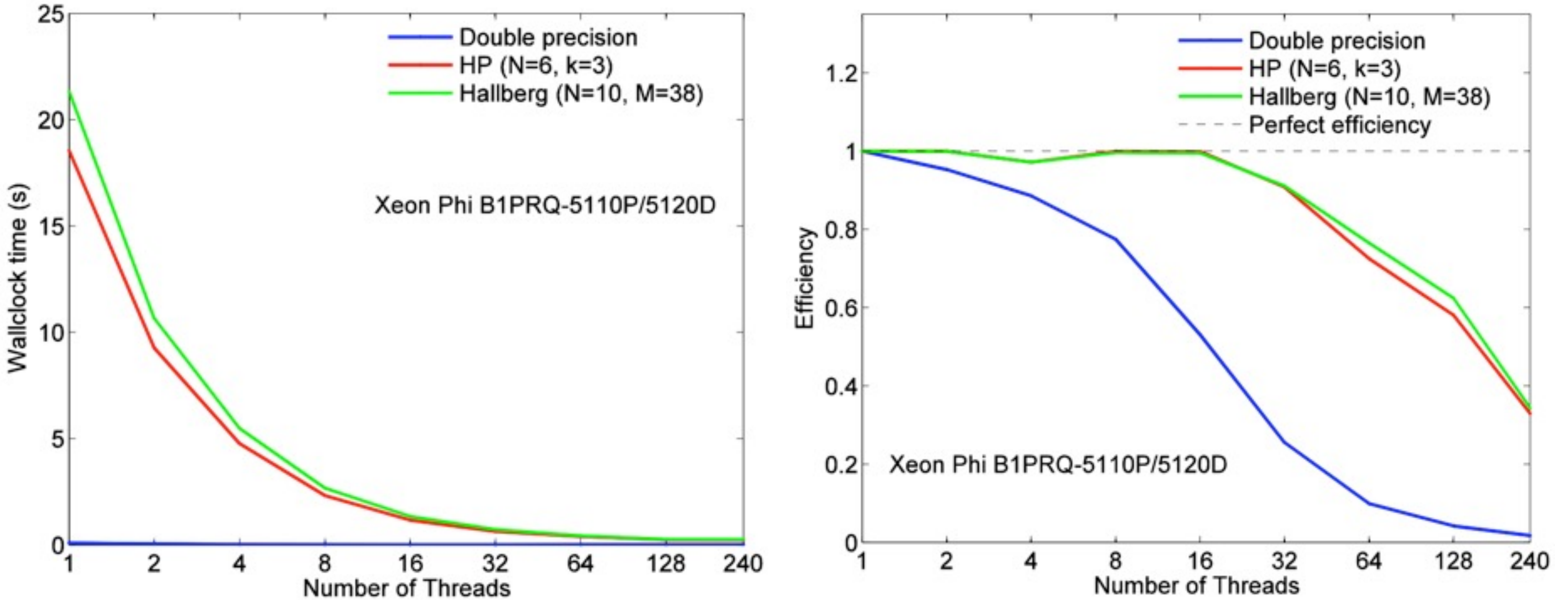


- **Faster speed of HP sum (7 reads & 6 writes on global memory) over Hallberg sum (11 reads & 10 writes)**

Parallel Efficiency on Xeon Phi

- **Runtime & strong-scaling parallel efficiency of HP, Hallberg & (order-sensitive) double-precision sums as a function of the number of threads on Intel Xeon Phi co-processor**

32 million summands



- **Faster speed of HP sum over Hallberg sum**

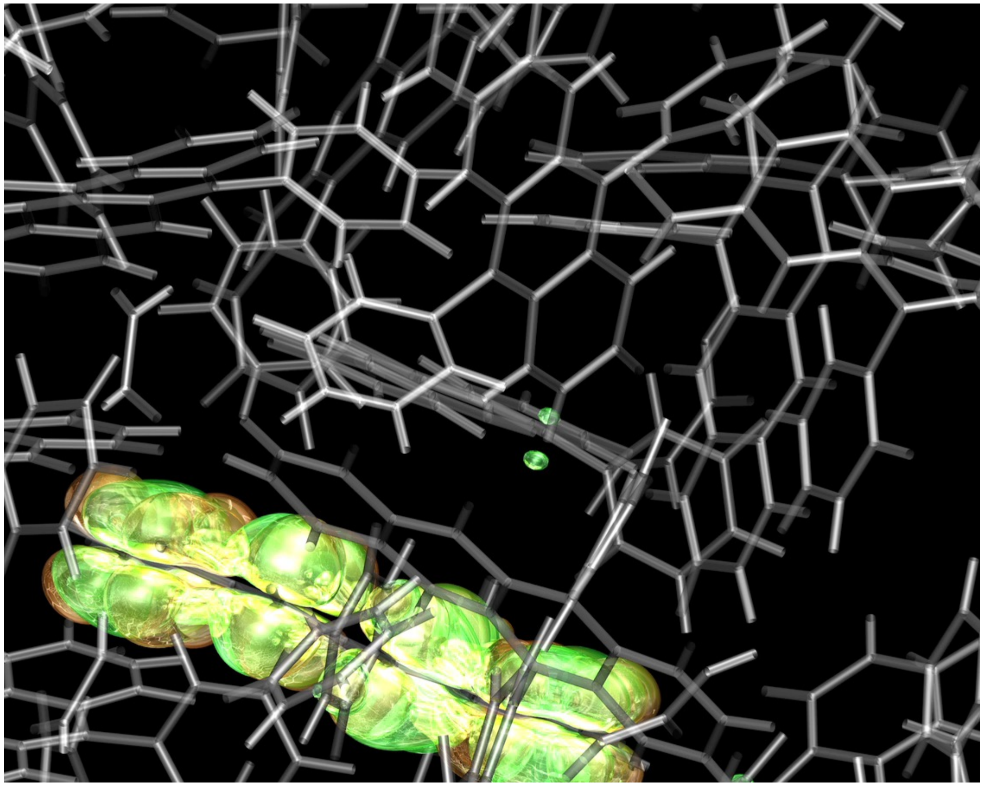
Large Production Simulations

- **16,661-atom quantum molecular dynamics (QMD) simulation on 786,432 IBM Blue Gene/Q cores suggests a rapid H_2-production technology that is industrially scalable**

  21,140 time steps (129,208 self-consistent-field iterations);
  Nano Lett. **14**, 4090 ('14)

- **Up-to 6,400-atom divide-conquer-recombine nonadiabatic QMD simulation reaches experimental time scales from first principles for photoexcitation dynamics**

  Appl. Phys. Lett. **102**, 173301 ('13);
  Sci. Rep. **5**, 19599 ('16)

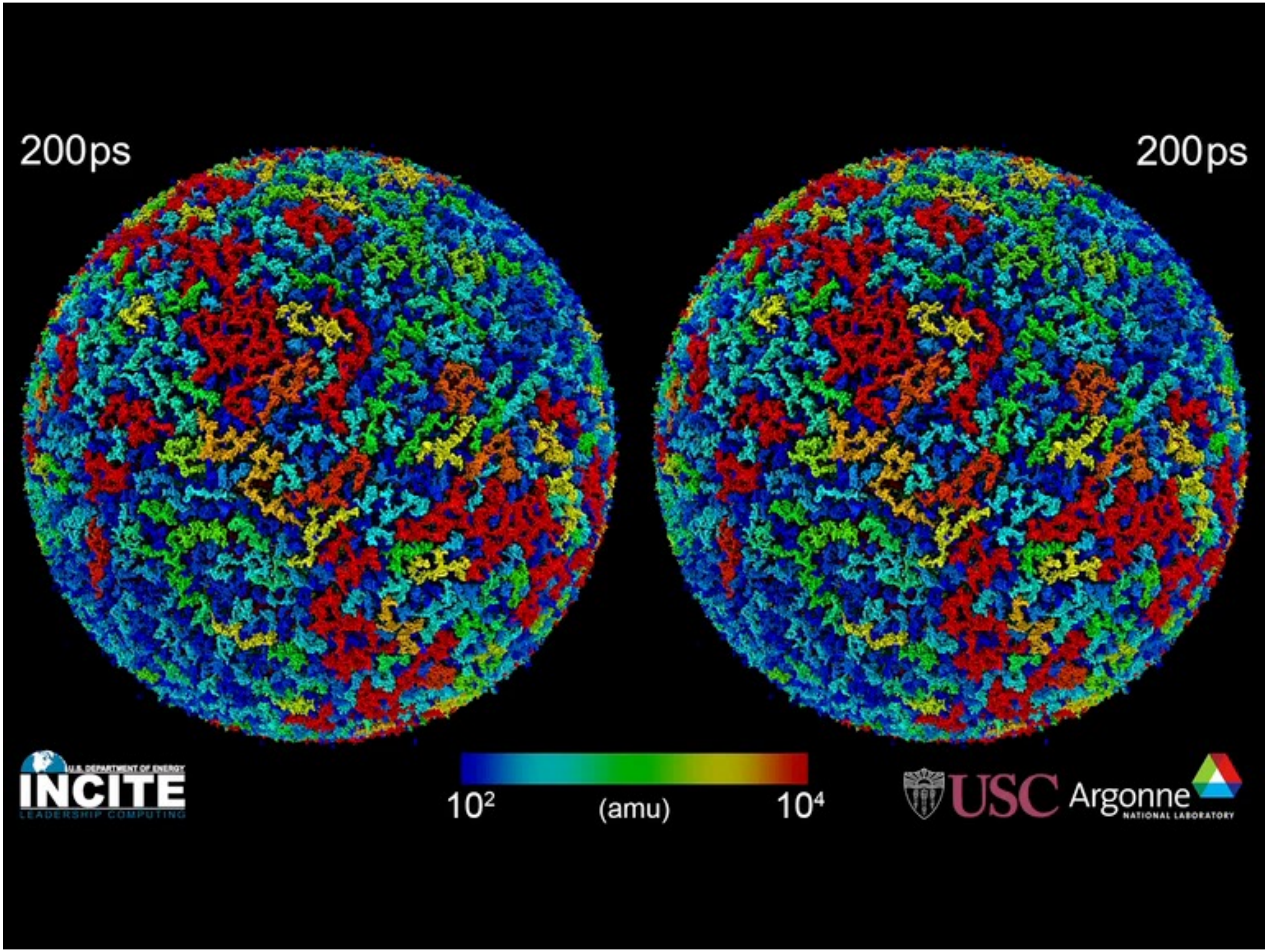


Quasi-electron Quasi-hole

- **112 million-atom reactive molecular dynamics (RMD) simulation on 786,432 IBM Blue Gene/Q cores reveals a simple synthetic pathway to fractal graphene**

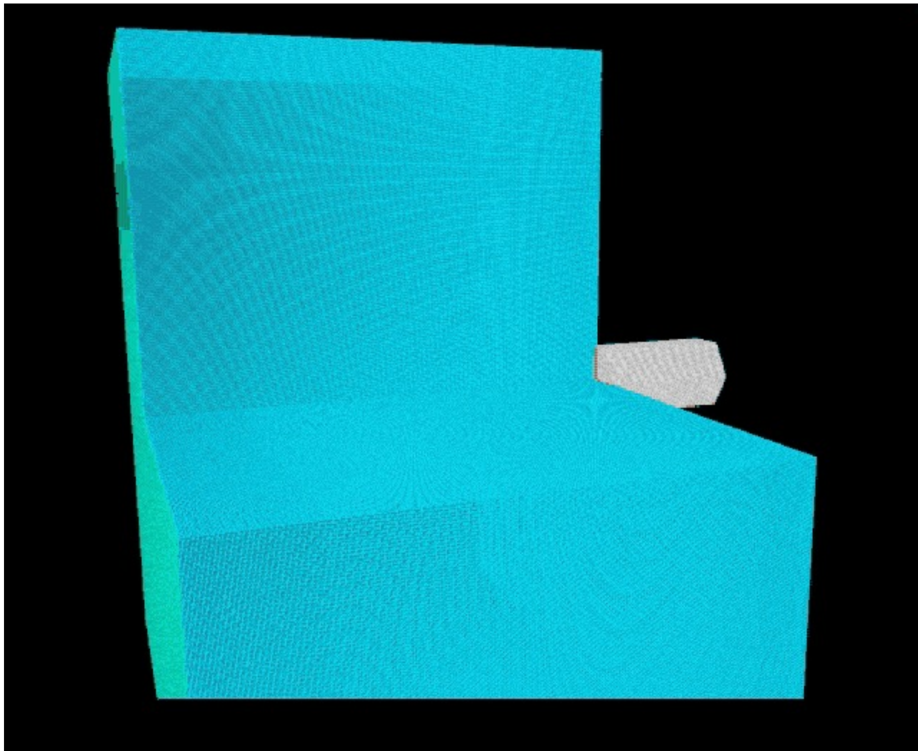
  Sci. Rep. **6**, 24109 ('16)

Percolation Transition



Movie made by J. Insley (Argonne)

Billion-Atom Molecular Dynamics

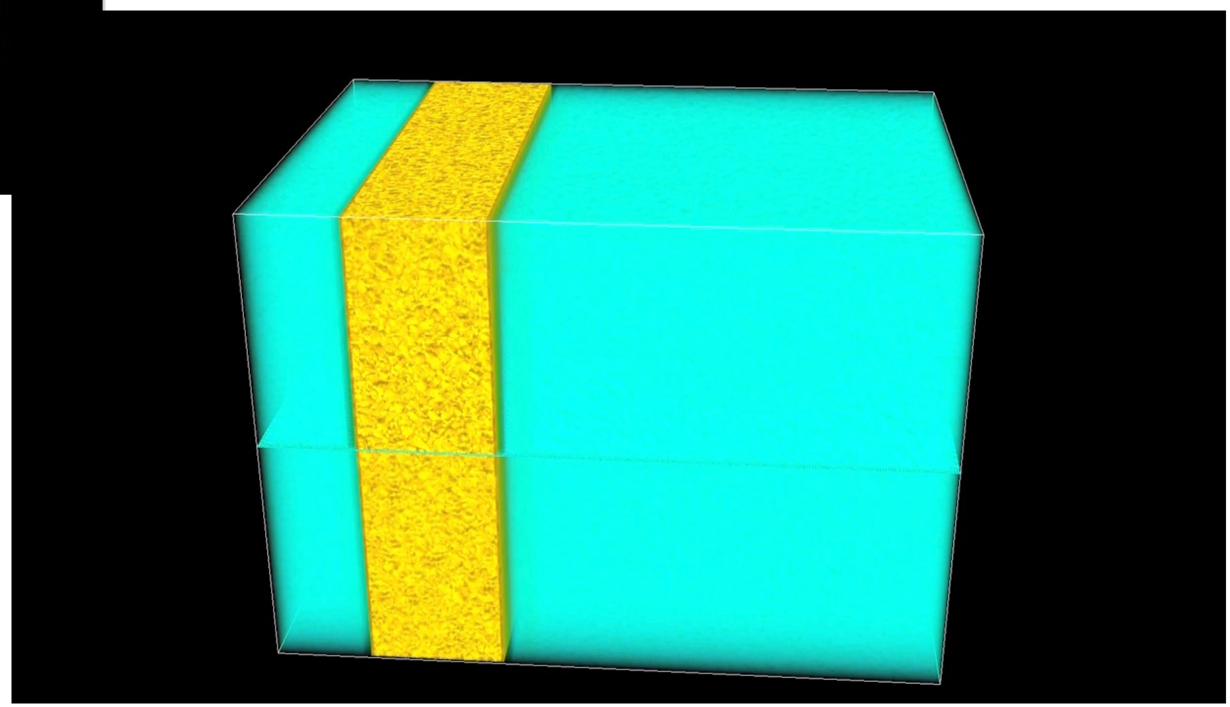


- **Shock-induced nanobubble collapse in water near silica surface (67 million core-hours of computing on 163,840 Blue Gene/P cores)**

A. Shekhar *et al*., *Phys. Rev. Lett*. **111**, 184503 ('13)

- **Hypervelocity impact on AlN**

P. S. Branicio *et al*.,
Phys. Rev. Lett. **96**, 065502 ('06)



100 nm

Conclusion

1. An order-invariant real-number summation method has been proposed for reproducible parallel computing
2. The proposed method achieves higher computing speed than the previous state-of-the-art for million+ summands on various parallel systems (MPI, OpenMP, CUDA, Xeon Phi)

Thank You

Research supported by
DOE Grant DE-SC0014607