Optimizing Stencil Computations: Multicore-optimized wavefront diamond blocking on Shared and Distributed Memory Systems

Tareq Malas[1], Hatem Ltaief[1], Georg Hager[2], Gerhard Wellein[2], and David Keyes[1]

[1]Extreme Computing Research Center - King Abdullah University of Science and Technology

[2]Erlangen Regional Computing Center (RRZE)

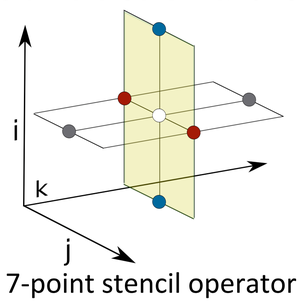
STENCIL COMPUTATIONS

- Bottleneck and time dominant in widely used scientific codes
- Appear in finite difference, element, and volume discretizations of PDEs
  - Example: $\frac{\partial^2 u}{\partial t^2} = \nabla \cdot \left(\alpha(x) \nabla u \right)$

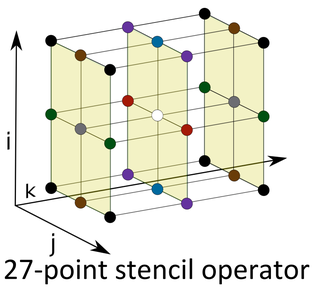
Example: 7-point stencil with constant coefficients

```
for t in N_t

  for i in N_z
    for j in N_y
      for k in N_x
        U(i,j,k)= C_0 * V(i  , j  , k  )
                + C_1 * V(i+1, j  , k  )
                + C_1 * V(i-1, j  , k  )
                + C_2 * V(i  , j+1, k  )
                + C_2 * V(i  , j-1, k  )
                + C_3 * V(i  , j  , k+1)
                + C_3 * V(i  , j  , k-1)
  V = U
```



7-point stencil operator

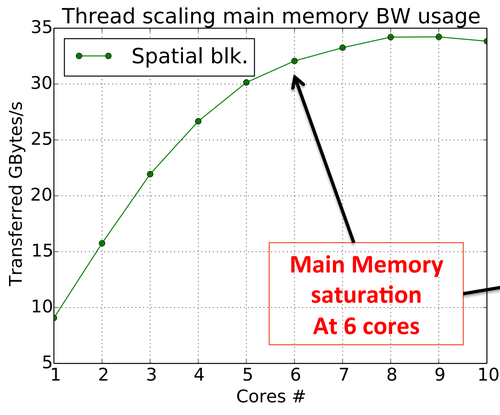


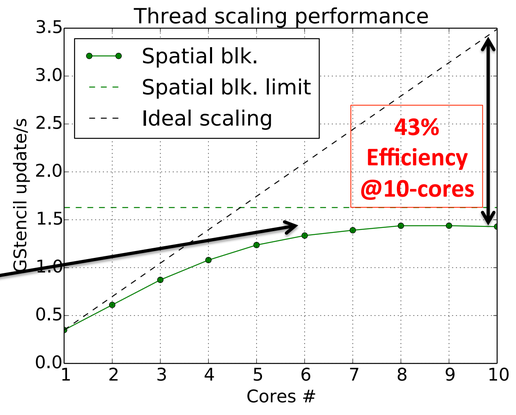
27-point stencil operator

CHALLENGES IN FUTURE ARCHITECTURES

- Each node may have up to a thousand shared-memory cores with
  - small cache size per core
  - small memory bandwidth per core
  - complex cache sharing among cores
  - expensive synchronization among all the cores
  - interaction between heterogeneous processors
- Expensive lock-step synchronization after each iteration

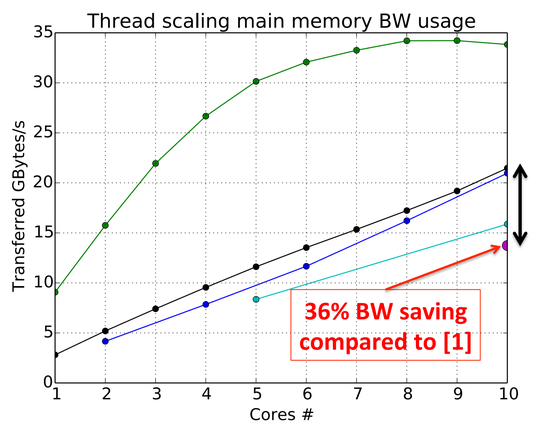
7-point stencil with constant coefficients

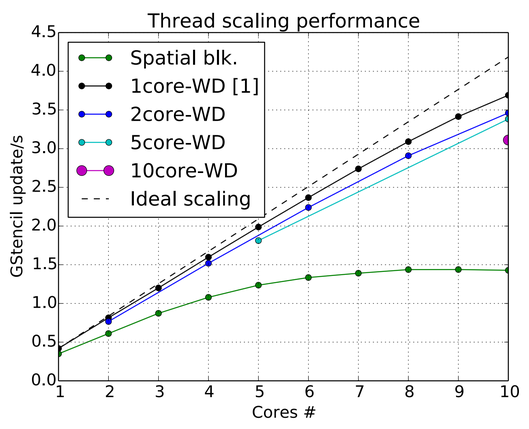




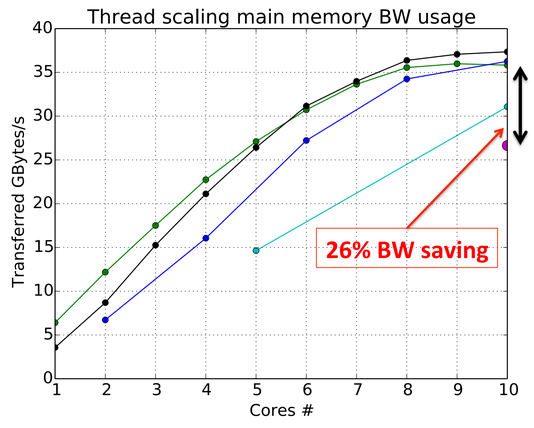
BREAKING MEMORY WALL THROUGH ADVANCED TEMPORAL BLOCKING TECHNIQUES

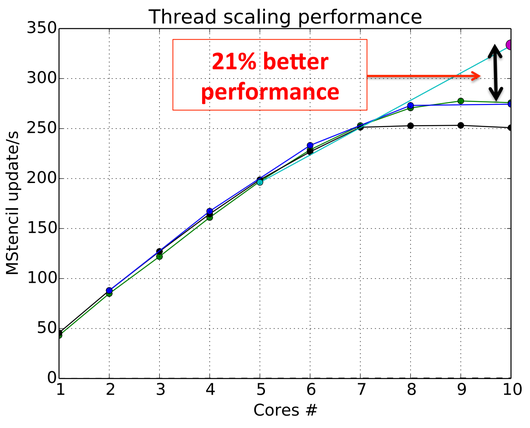
7-point stencil with constant coefficients





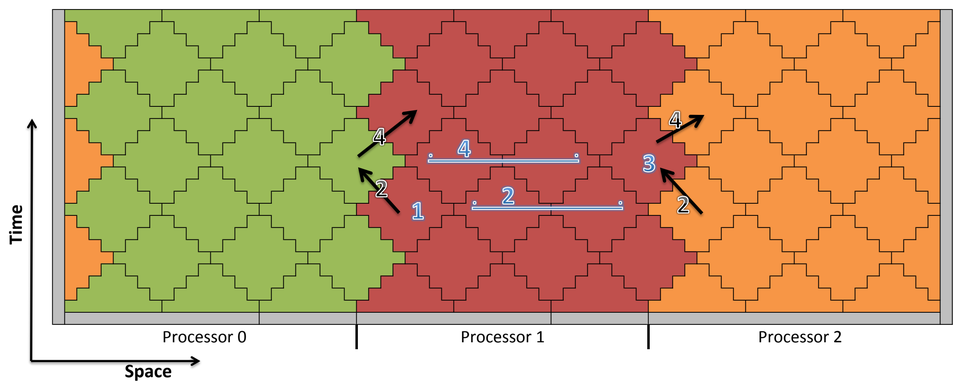
25-point stencil with variable coefficients





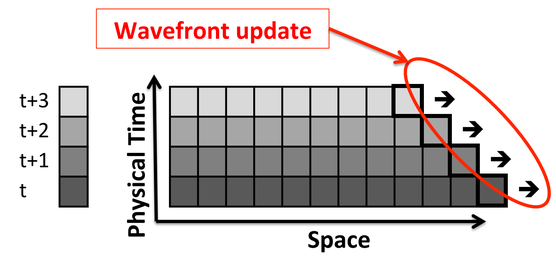
DIAMOND TILING FOR SHARED AND DISTRIBUTED MEMORY SYSTEMS

- High data reuse, reducing memory accesses
- Provides independent space-time blocks to
  - reduce synchronization between threads and nodes
  - overlap computation with communication
- Assignment of diamond tiles to Thread Group (TG):
  - Within diamond: tight synchronization and large cache blocks
  - Across diamonds: loose synchronization between TGs
- Can provide efficient data migration for neighbor processes load-balancing
- Tessellation reduces the overhead of handling the boundaries of subdomains, sockets, and heterogeneous processors

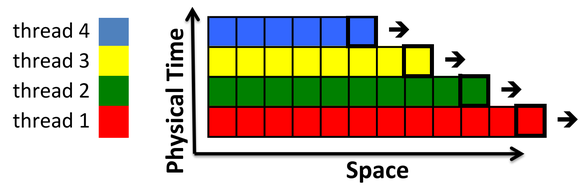


WAVEFRONT TEMPORAL BLOCKING

Single-core wavefront increases data reuse in the CPU cache.



Multi-core wavefront increases data reuse among CPUs caches, with the advantage of larger cache block per core.

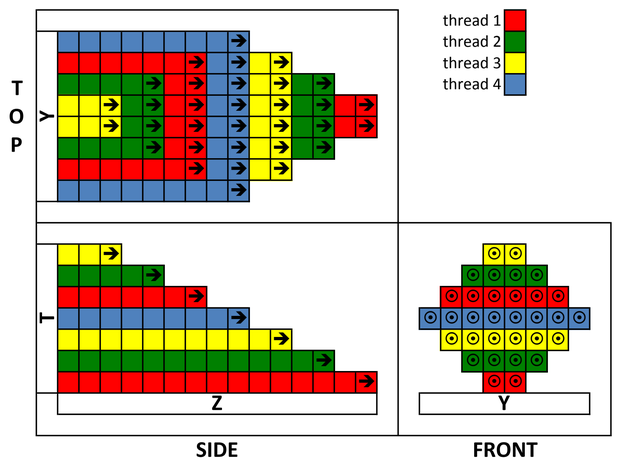


APPROACH: DIAMOND TILING + WAVEFRONT TEMPORAL BLOCKING

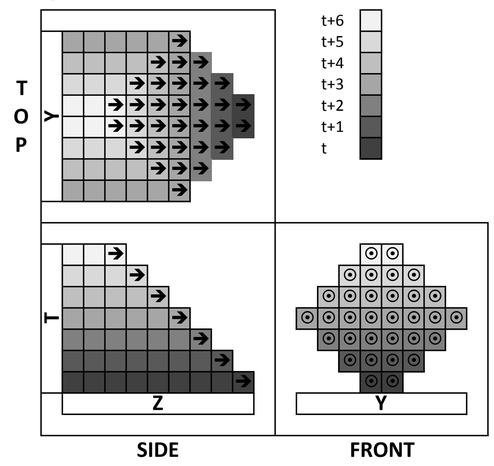
Extruded diamonds

- Diamond tiling and domain decomposition across the Y-axis
- Wavefront temporal blocking along the Z-axis
- No decomposition across the X-axis

Multi-core wavefront diamond



Single-core wavefront diamond [1]



SUMMARY OF SIGNIFICANCE

- Large reduction in cache block size and memory bandwidth requirements
- Utilizes the shared cache between cores of modern processors
- Controllable tradeoff between memory per thread and frequency of synchronization
- Relaxed synchronization of MPI messages in distributed implementation
- Overlaps computations with communication in distributed implementations

REFERENCES

[1] Strzodka, R., Shaheen, M., Pajak, D., & Seidel, H.-P. (2011). Cache Accurate Time Skewing in Iterative Stencil Computations. *In International Conference on Parallel Processing* (pp. 571–581). doi:10.1109/ICPP.2011.47

Acknowledgements: Texas Advanced Computing Center (TACC) for hardware access
KAUST Extreme Computing Research Center for support of T. Malas