FWCG2013

Parallel Multiple Observer Siting on Terrain

Wenli Li, W. Randolph Franklin, Daniel Benedetti

Rensselaer Polytechnic Institute
Troy, NY, 12180

25 Oct 2013

Outline

- Multiple observer siting on terrain
- Parallel implementations
- Experiments

Multiple Observer Siting

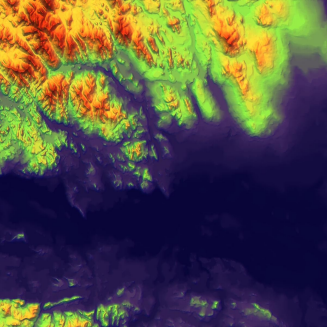
- Multiple observer siting on terrain is
  - to place a number of observers on a terrain, which cover as many targets on the terrain as possible
  - or to cover a number of targets using as few observers as possible
- Our assumptions
  - The terrain is a digital elevation model
  - The targets are the terrain points, raised to a certain height
  - The observers are placed at terrain points, raised to a certain height
  - A target is covered by an observer if it is visible from the observer, up to a certain radius of interest
- Implications
  - The targets covered by an observer constitute its <u>viewshed</u>
  - The targets covered by all the observers constitute their <u>cumulative viewshed</u>

Multiple Observer Siting

- The original siting package
  - was developed by Franklin and Vogt in 2006
  - has four programs that run in order: VIX→FINDMAX→VIEWSHED→SITE
- VIX computes an approximate visibility index for each point
  - Algorithm: for each point, pick some random targets and compute the percentage of the targets that are visible, as the approximate visibility index
  - Parameters: the number of rows and columns of the terrain, the radius of interest, the observer and target height, and the number of tests for each point
- FINDMAX finds some most visible points as tentative observers
  - Algorithm: divide the terrain into equal-sized blocks and find the same number of tentative observers within each block
  - Parameters: the side length of a block and the number of tentative observers per block

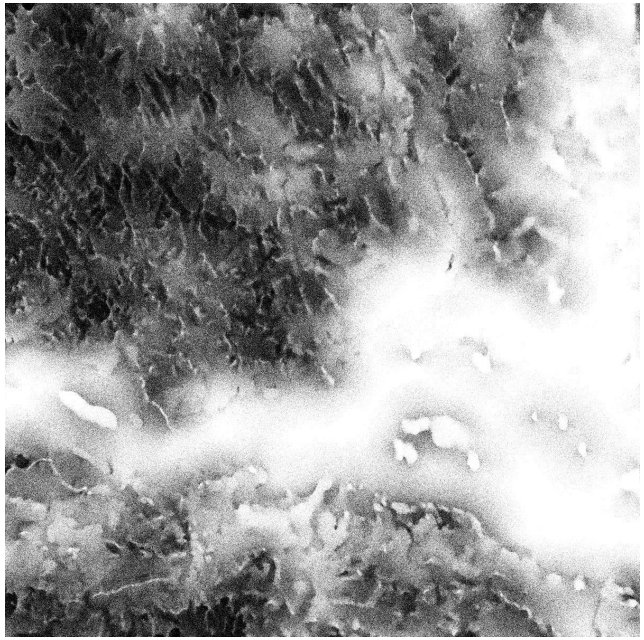
Multiple Observer Siting

- VIEWSHED computes the viewshed of each tentative observer
  - Algorithm: compute the visibility of points along lines of sight from the observer to each point around the perimeter of a square centered at the observer
  - Parameters: the radius of interest and the observer and target height
- SITE selects the observers to cover the terrain
  - Algorithm: incrementally select observers, one at a time, whose viewshed adds the most area to the cumulative viewshed
  - Parameters: the termination condition



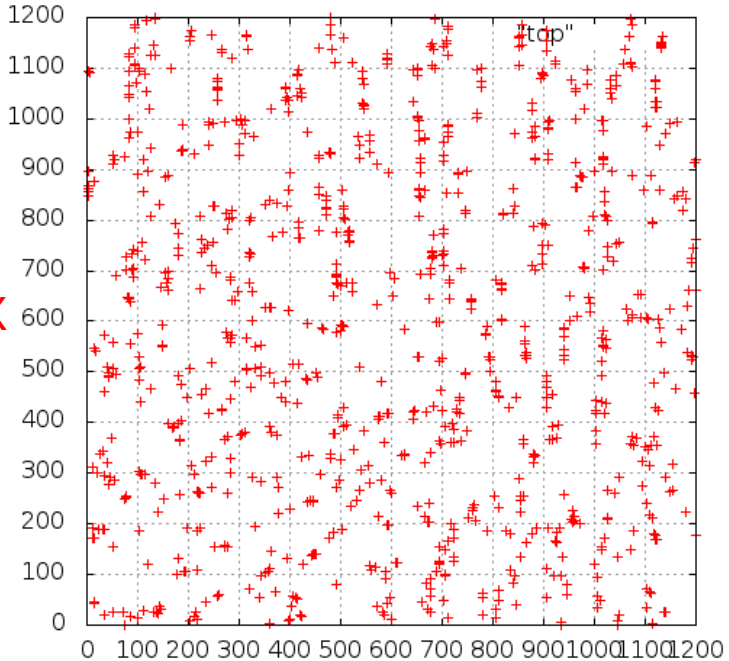
1201×1201 points

VIX

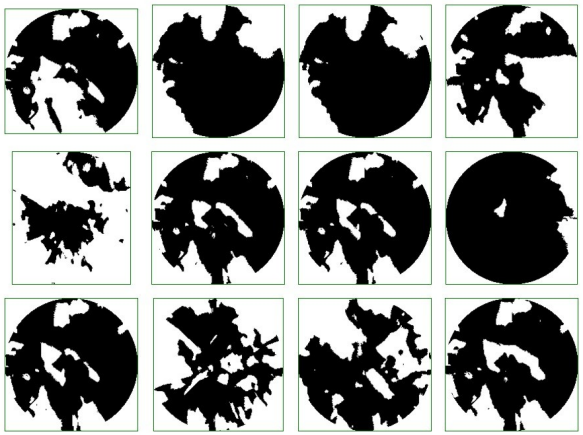


FINDMAX

12×12 blocks,
1008 tentative observers

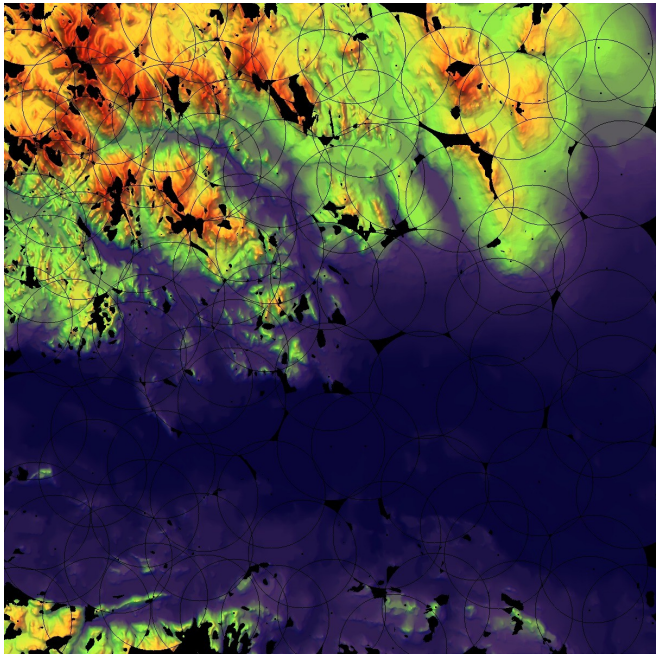


VIEWSHED



SITE

99 observers, 95%



Parallel Implementations

- Why parallel?
  - Multiple observer siting is compute-intensive and has much inherent parallelism
- Benefits of a faster package
  - Higher resolution terrains
  - More accurate visibility indexes
  - More tentative observers
- Two methods are used to parallelize the package
  - OpenMP, to utilize the many cores of multi-core CPUs
  - CUDA, to utilize the general purpose programming capabilities of NVIDIA GPUs

Parallel Implementations

- Changes to the sequential programs
  - Merging the four programs into one
    - Eliminating intermediate file I/O's
    - Four functions: VIX, FINDMAX, VIEWSHED, and SITE
  - An improved algorithm for SITE
    - Computing the extra area that would be added to the cumulative viewshed C by a tentative viewshed V as area(V – C) instead of area(Union(C, V)) – area(C)
    - Reducing the complexity from the size of the terrain to the size of a viewshed
    - Significant when the radius of interest is small

Parallel Implementations

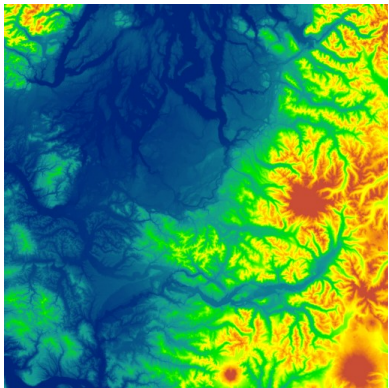
- The OpenMP program adds a few compiler directives to the sequential program
  - In VIX, a `#pragma omp parallel for` before the for loop that computes the approximate visibility indexes
  - In FINDMAX, a `#pragma omp parallel for` before the for loop that finds the most visible points
  - In VIEWSHED, a `#pragma omp parallel for` before the for loop that computes the viewsheds
  - In SITE, a `#pragma omp parallel for` before the for loop that computes the extra areas

Parallel Implementations

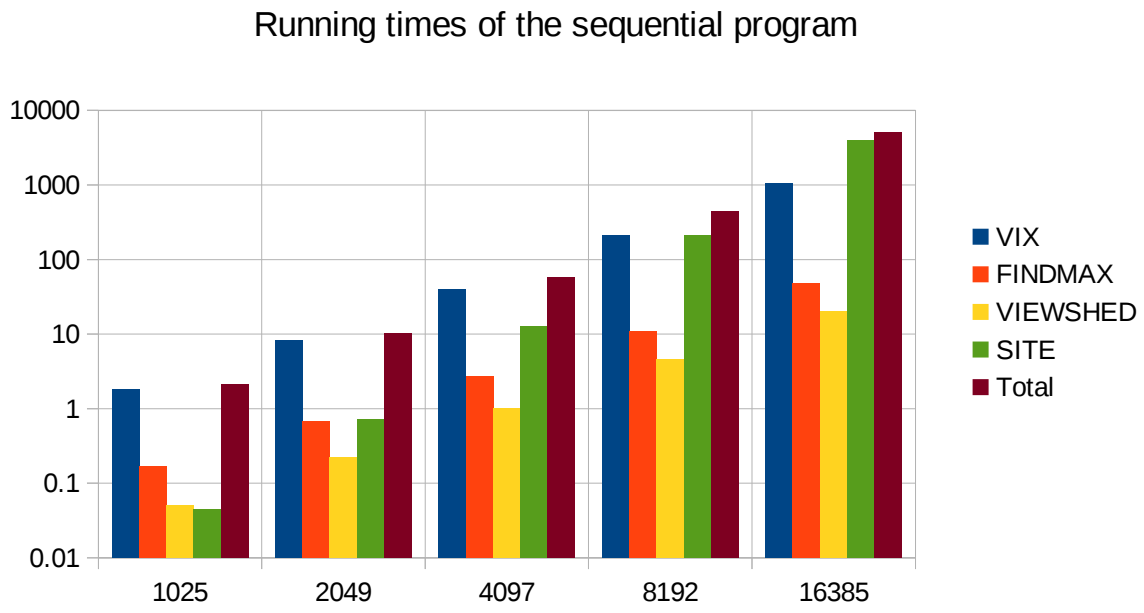
- The CUDA program defines a kernel function for each of the four parts
  - In VIX, a CUDA thread computes the approximate visibility index of a point
  - In FINDMAX, a thread block sorts the points of a terrain block and the first thread copies the top points
    - A thread sorts a fraction of the points
  - In VIEWSHED, a thread block computes the viewshed of a tentative observer
    - A thread computes a sector of the viewshed
  - In SITE, a thread block computes the extra area of a tentative viewshed
    - A thread processes several rows of the viewshed

Experiments

- The test machine
  - Dual Intel Xeon E5-2687 CPUs with 16 cores and 32 threads
  - 128GB of memory
  - NVIDIA Tesla K20x GPU accelerator with 2688 CUDA processing cores and 6GB of memory
- The test terrains
  - 1025 × 1025, 2049 × 2049, 4097 × 4097, 8193 × 8193, 16385 × 16385 Puget Sound terrains
  - Derived from a 16385 × 16385 Puget Sound terrain of P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. *Proceedings of the conference on Visualization '01*

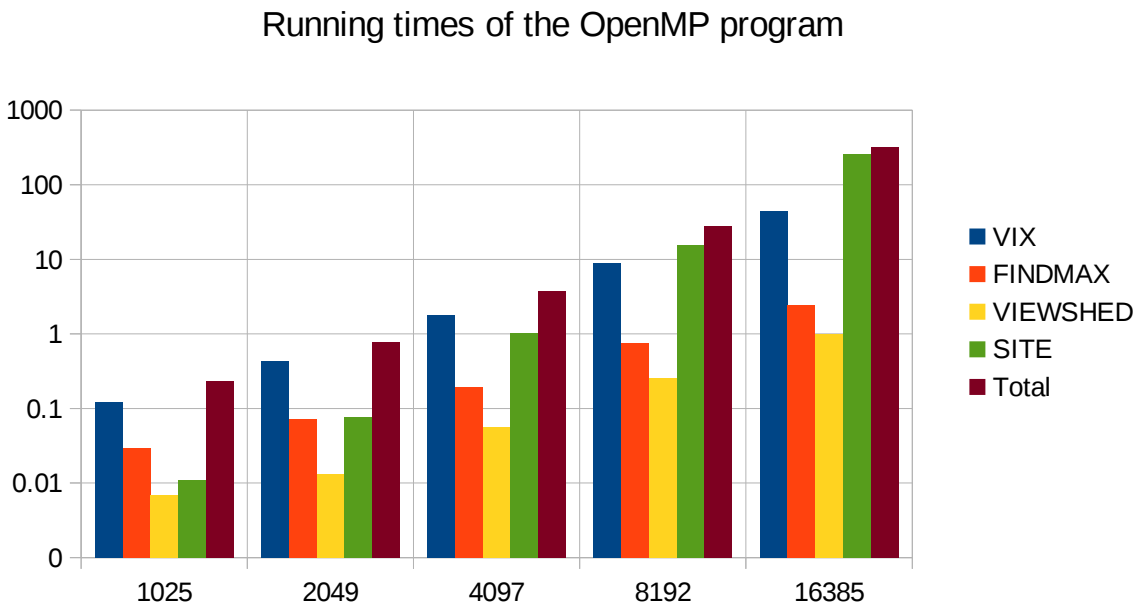


Experiments



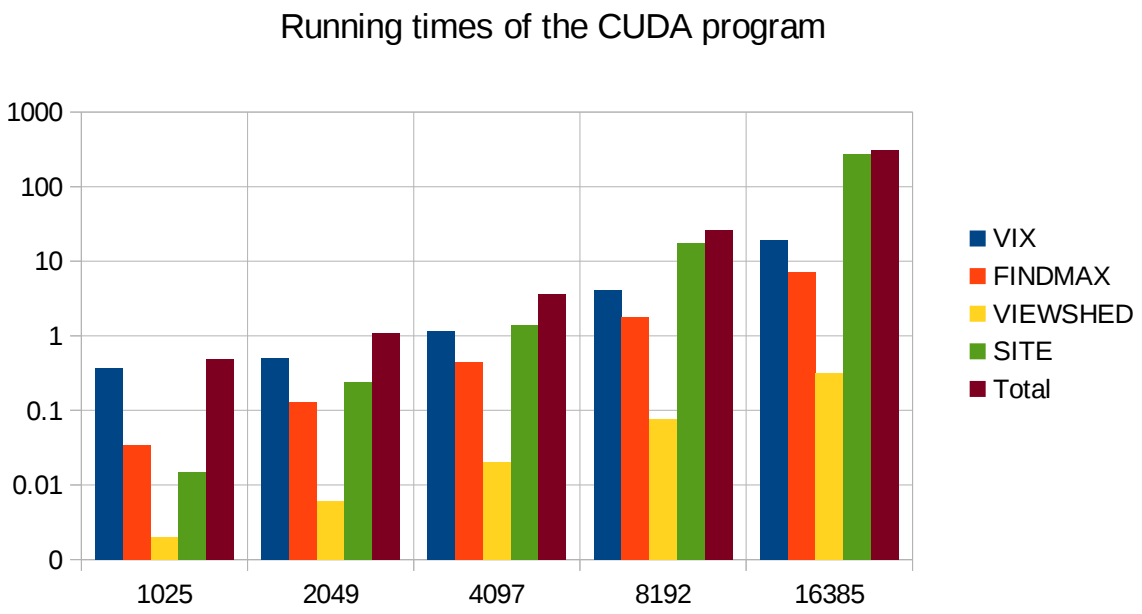
- Parameters: radius = 50 points, height = 2 meters, VIX tests = 10, block side = 100 points, one tentative observer per block
- The total number of tentative observers are 100, 400, 1681, 6724, 26896
- VIX and FINDMAX are roughly linear to the terrain size
- VIEWSHED is linear to the total number of tentative observers
- SITE is roughly linear to the square of the number of tentative observers

Experiments



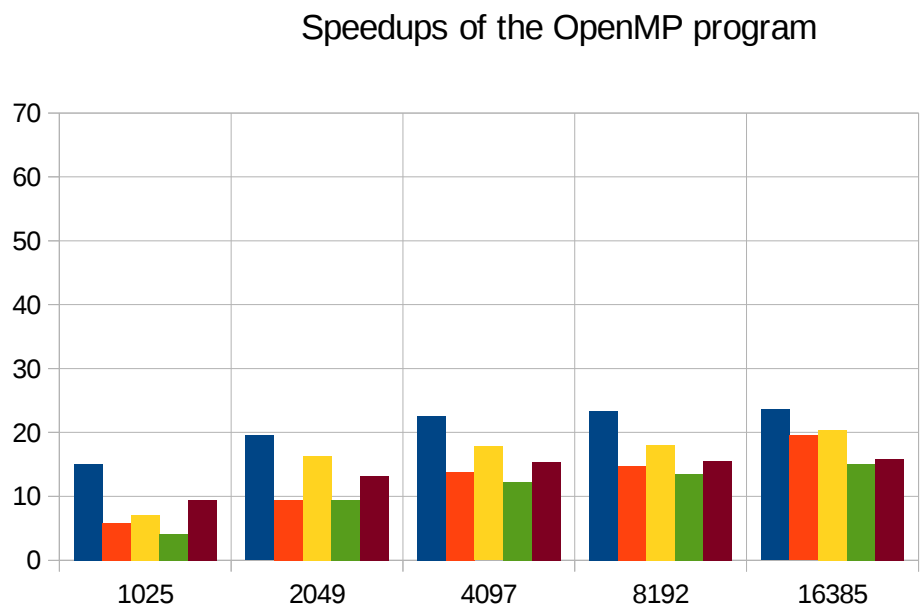
- The OpenMP program is much faster

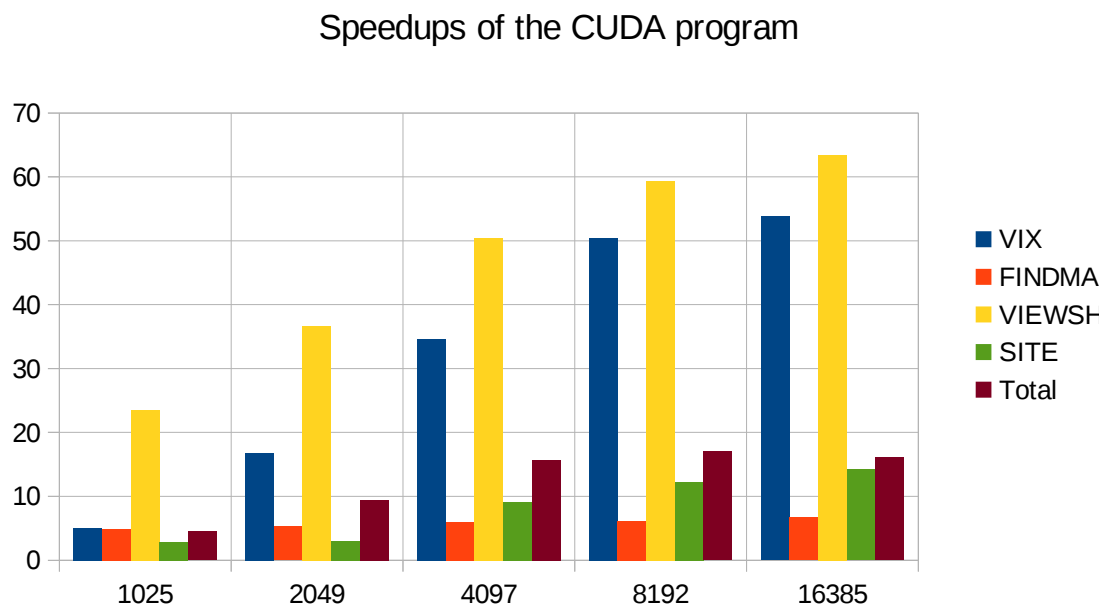
Experiments



- The execution configuration has a great impact on performance
- We tried 32, 64, 128, 256, 512 and 1024 as the number of threads per thread block and selected (128, 512, 512, 32) for the four parts
- SITE and total time are similar to the OpenMP program
- VIX and VIEWSHED are faster
- FINDMAX is slower

Experiments

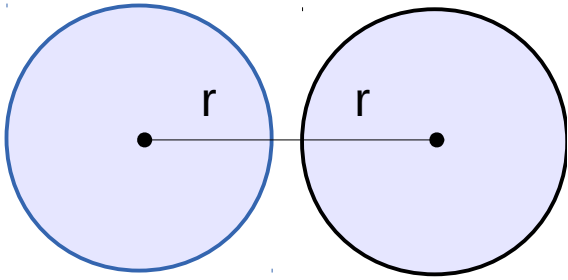


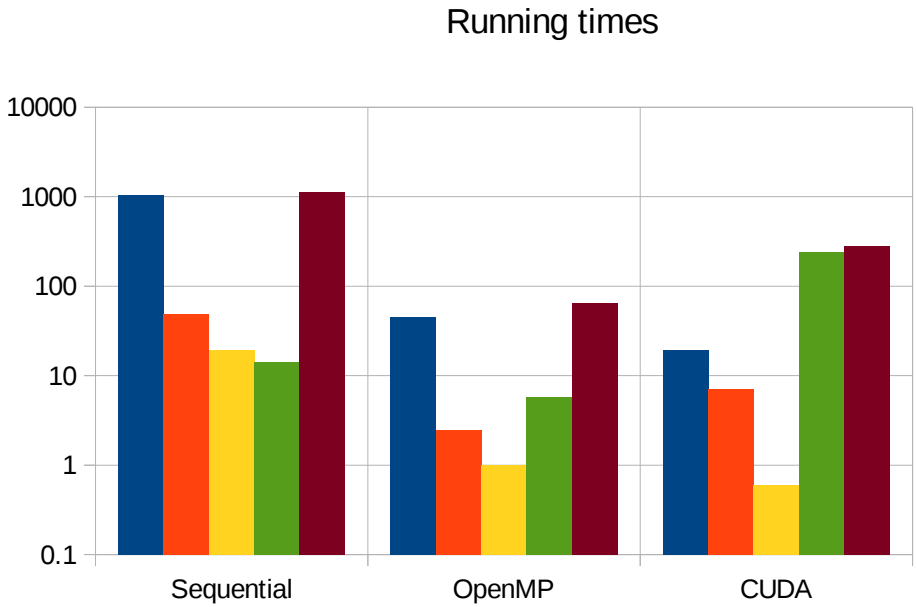


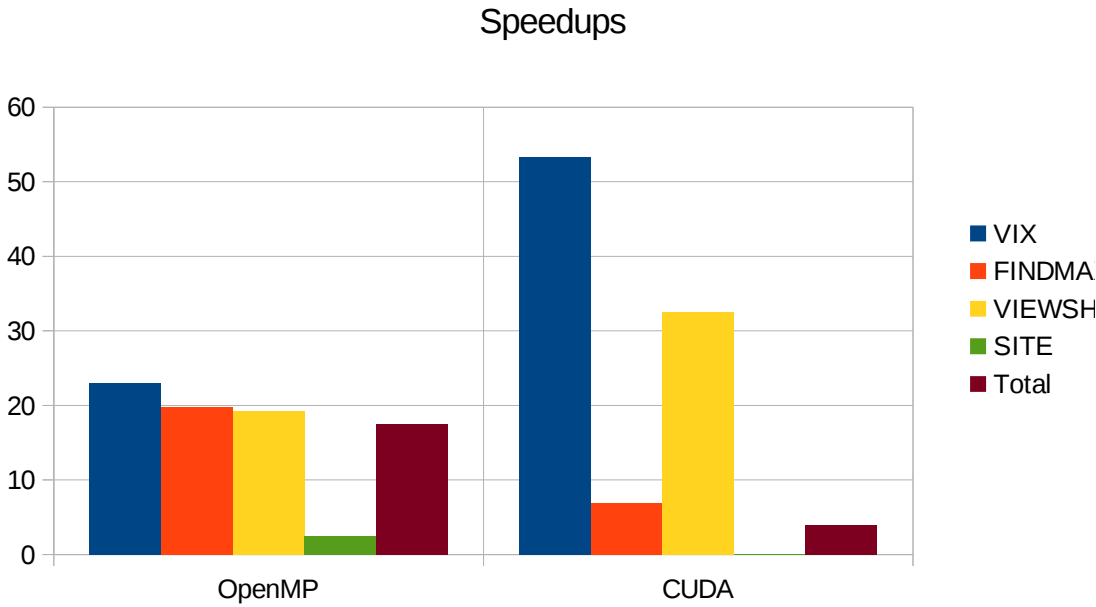
- Although the speedups of the total time are similar for larger terrains, the speedups of the four parts are quite different
- The CUDA program is much faster in VIX and VIEWSHED: a high degree of parallelism
- Much slower in FINDMAX: lack of parallelism in sorting; inefficient implementation
- A little slower in SITE: overhead of a kernel launch in each iteration; inefficient implementation

New Results

- When the radius of interest is small, there is no need to recompute all the extra areas in each iteration
- Only compute the ones that may change
- SITE is no longer a time-consuming part
- There is not much parallelism in SITE and the running time of CUDA SITE is dominated by overhead







Conclusions

- Siting has much inherent parallelism and both OpenMP and CUDA are effective ways to parallelize the program
- The OpenMP program is easy to implement
- The CUDA program is hard to implement and configure, but could have more potential
- Future work
  - More efficient implementations
  - More parallelism in SITE: to site multiple observers per iteration
  - To combine OpenMP and CUDA