Parallel FlowViz

Angel Manica V. Raquel and Joseph Anthony C. Hermocilla

***Abstract*— This project improved FlowViz, a flood simulation software. It applied the concepts of parallel computation to adapt for wider area simulations. Using Data Elevation Models or DEMs as data input this program computes for the possible catchment areas using the Triangular Multiple Direction (MD∞) water delineation method and outputs a 3D visual representation of the waterflow. It was designed to use parallel computation where the master node was assigned for the division of data into subgrids which can then be processed by separate nodes locally and then sent back to the master node for the visual representation of the data.**

***Index Terms*— Parallel Computing, DEM, Triangular Multiple Flow Direction Algorithm**

I. Introduction

A. A. Background of the Study

The main aim of this project is to improve FlowViz , a flood simulation software [1] . The data used in FlowViz was limited to only the mountain areas of Luzon and this project will aim to use a larger scope of location in every place in the Philippines with available Digital Elevation Models(DEMs). Flood simulation involves a great deal of computation and with a larger data set, this project seeks to reduce the computation time by modifying FlowViz to apply parallel computation for a wider area simulation.

B. B. Statement of the Problem

A flood is a hydrological event characterized by high discharges and/or water levels that can lead to inundation of land adjacent to streams, rivers, lakes, wetlands and other water bodies. It is the costliest natural hazard in the world. It can cause damage to property and/or human lives. What is more, there has been an increasing trend associated to the number of deaths caused by flood [2]. Since floods are mostly caused by natural factors, the extent of their damages can be difficult to predict, but with relevant information, planning ahead can be done to minimize the damages they may bring.

C. C. Significance of the Study

Flood models are used to simulate how rain water flows overland to determine the probability of flooding [3]. They can also be used in planning and designing structures so that pre-emptive measures can be taken in time . Early flood warning systems serve to save lives, minimize flood damage to properties and reduce economic and social losses [4]. Awareness in the probability of flooding in an area should give time to construct temporary defences, to evacuate people if necessary and to move valuables from buildings and properties. Also, not all floods are harmful especially those that are small enough to just create wetlands which can be useful in agriculture. FlowViz is software developed to model the surface water flow over a wide area based on digital elevation models. Being able to model the surface water flow will allow the identification of possible catchment areas where flood might occur in the future. This study will aim to improve FlowViz by using a different method in catchment computation and by incorporating the concept of parallel computing.

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

D. D. Objectives of the study

The main aim of this project is to improve FlowViz. Specifically, the study aims to:

- Use parallel computation in the simulation,
- Improve the Visualization of the Study Areas, and
- Implement the triangular multiple flow direction (MD∞) algorithm.

E. E. Date and Place of Study

This study was conducted in the Institute of Computer Science, College of Arts and Sciences, UPLB from November 2011 to March 2012. Using the MPC cluster, one of the Grid services provided by the Advanced Science and Technology Institute (ASTI) High-Performance Computing (HPC) Facility and Philippine e-Science Grid (PSciGrid).

II. Related Work

Through Digital Elevation Models (DEMs), hydrology determines the path of water/flow directions. One of the earliest methods in tracing water flow is the D-8 algorithm, which was introduced by OCallaghan and Mark (1984). This Method was implemented in FlowViz [1]. Several grid-based flow routing algorithms have already been presented by different authors after the development of D8 (Quinn et al. 1991, Freeman, 1991; Holmgren 1994; Mitasova and Hofierka, 1993; Mitasova et al. 1995, 1996; Tarboton, 1997). The triangular multiple direction (MD∞) algorithm presented by Seibert and McGlynn (2007) combines the benefits of triangular single direction flow (D∞) and the other multidirectional flow algorithms. It is more appropriate than the other existing flow algorithms across a range of landscapes, DEM resolutions and applications. FlowViz was originally implemented as a sequential program, and so as many other currently existing programs. Although the speed at which sequential computers operate has been improving at an exponential rate for many years, the improvement is now coming at greater and greater cost. As a consequence, researchers have sought more cost-effective improvements by building parallel computers which

perform multiple operations in a single step [5]. Around 1980, people believed the use of more efficient processors will best improve a computers execution but that idea was disputed by the concept of parallel processing which consequently lead to making the use of clusters of computers or work stations as the standard platforms for high performance and large-scale computing [6]. In 2006, Pabico presented the analytical model of the parallel implementation of the D8 flow routing algorithm; performance metrics such as parallel speed up, parallel efficiency, parallel cost, cost optimal function, and scalability, under the Parallel Random Access Memory Model were presented [7]. In a world where parallel processing is being enhanced, more and more programs are being designed to use parallel computing especially to problems which involves complex computations and exhaustive memory access [6]. There had also been a study conducted in the Institute of Computer Science (ICS) in UPLB for Parallel Stream Delineation where the triangular single direction flow (D∞) routing algorithm was also implemented as a parallel program to trace the flow of water in a terrain, where the program was successfully performed in DEMs and node clusters of different sizes [8]. Previous works thus indeed show that the parallelization and improvement of FlowViz is possible.

III. THEORETICAL FRAMEWORK

A. Digital Elevation Models (DEMs)

The project will make use of DEMs as data input to compute for the catchment areas. A Digital Elevation Model (DEM) is a digital representation of a ground surface elevation, It is a grid of points that can be defined by X and Y coordinates [9].

B. Probability of Precipitation (POP)

Probability of Precipitation (PoP) is the likelihood of measurable precipitation at a particular point during a specified time period. It is stated as a percentage and is also referred to as chance of rain or chance of precipitation. Represented as a grid where the probability of precipitation is specified in each cell. Each elevation cell in the DEM will have a corresponding cell in the POP matrix.

C. Triangular Multiple Direction Flow (MD∞) Algorithm

The project will need to be able to trace the flow using a process that will involve the idea that the water from a higher grid of elevation will flow to an area of lower elevation. FlowViz made use of the eight-direction pour point algorithm or Deterministic 8 model (D-8) to determine the flow direction [10]. It assigns a flow direction code to each cell, based on the steepest downhill slope as defined by the DEM. It is a simple and traditional algorithm which is why it is more commonly used. D-8 is limited due to obvious reasons-It restricts the flow of water to only one direction against eight and therefore somewhat unrealistic.

In April of 2007, Seibert and McGlynn introduced a new method for computing upslope areas from gridded digital elevation models, the triangular multiple flow direction algorithm [11]. This project will aim to use this method. Around the midpoint (M) of the pixel in question, eight planar triangular facets are constructed with midpoints (P1 and P2) of two adjacent pixels (Figure 3). The slope direction of each of these triangular facets is then calculated. For directions pointing between P1 and P2 the flow is distributed to the two cells on the basis of the direction of the steepest slope.

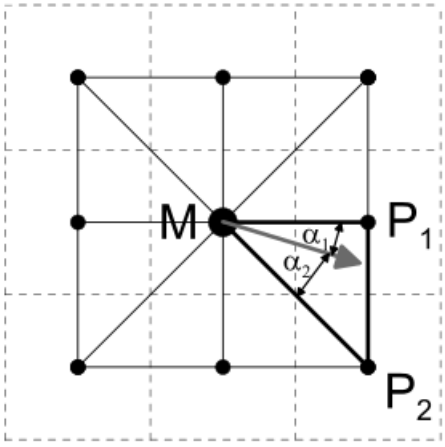


Fig. 1. Illustration of how the neighboring nodes were paired [11].

D. Parallel Computing

Parallel computing in simple terms is using more than one computing processor to solve one problem. This is accomplished by breaking the problem into independent parts so that each processing element can execute its part of the algorithm simultaneously with the others. It is very useful in problems which involve complex computations [12].

IV. MATERIALS

Software Interface

- GLUT/xf86vmode (OpenGL Utility Toolkit)
- GDAL (a software library that makes it easy to read and write raster geospatial data formats)
- GCC (C source code compiler)
- MPICH (provides interfaces to send/receive data and synchronise operations between the multiple tasks of a parallel application)

V. METHOD

A. Cluster Setup

A cluster of computers from DOST-ASTI(Advanced Science and Technology Institute) was used for this study. The cluster can have up to 30 nodes but they are not all readily available unless requested with each node having the standard MPICH installed so as GDAL. The cluster was accessed using Secure Shell which is a network protocol that allows data to be exchanged using a secure channel between two networked devices without the requirement of a password. The cluster of computers was setup using a master-slave network configuration by the ASTI staff.

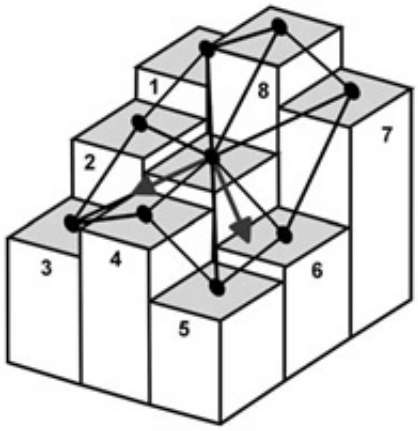


Fig. 2. Illustration of how the water flow was distributed [11].

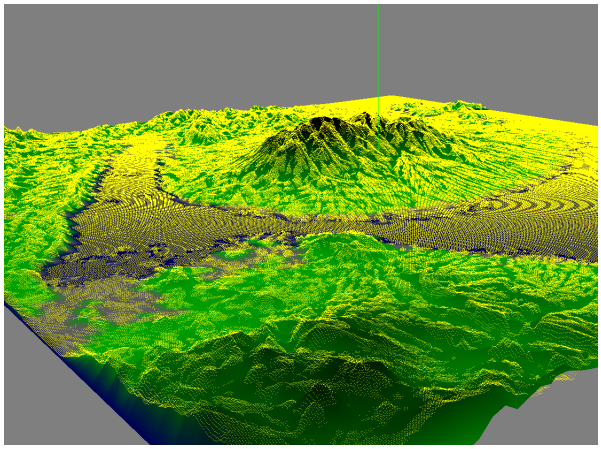


Fig. 3. 3D rendered image where each node has 1 as value of rainfall.

B. Data Management

The data extracted from the DEMs were partitioned into subgrids depending on the number of processors required by the current run from the cluster. The data is extracted in the form of a one-dimensional array. The subgrids were then distributed to the nodes where each node will perform the water flow computation on their local grids. These local girds were partitioned in such a way that the data is overlapping so that in the computation of the traceflow, it will be as though the data is continuous and not partitioned. The approach used in this study for the data partition was that the data was initially divided into half and then was further divided until the data division reaches the desired number of partition depending on the number of nodes. Simple blocking point-to-point communication was used throughout the study of the communication of the nodes using the commands MPI_Recv and MPI_Send.

The data was sent as an array over the cluster and it's the job of the slave nodes to convert the allotted arrray sent to them to a structure which will then be later used for the computation. For each of the cell in the local subgrid, the flow direction was computed for each of the neighboring cells. Triangular facets (phases) were used for the computation for each of the neighboring nodes. For each cell in the localgrid, the values of its neighboring nodes were gathered and stored as an array. The array of the current cell was passed to a function which computes for the steepest direction using the triangular multiple flow direction algorithm.

C. Local Flow Trace Computation

Each of the neighboring cells was paired to its adjacent node which will form one of the eight triangular facets with the midpoint before the computation (see Figure 1 on page 2).

Around the midpoint (M) of the cell, eight planar triangular facets are constructed with midpoints (P1 and P2) of two adjacent neighboring cells. For each of these local planes the direction of the steepest gradient is computed.

If this steepest direction from M is outside the 45 (p/4 radian) angle range of the particular triangular facet (i.e., not between the vectors pointing from M toward P1 and P2, respectively), the direction with the steeper downslope gradient, of the two directions toward P1 or P2, is used as the steepest direction, and the slope is computed between M and P1 or P2. If both P1 and P2 have higher elevations than M, both directions, obviously, are excluded. After computing the steepest downslope directions for all eight triangular facets, those directions are maintained as the locally steepest directions that are within the 45 angle range.

After the downslope directions are all computed for a cell, its accumulated area is distributed to these directions. Accumulated area is weighted and distributed downslope on the basis of the gradients. If a downslope direction falls between two neighboring cells, the area is further distributed to these two cells according to the relative differences in their direction.

D. Catchment Computation

After detemining the directions of the flow for each element, the next step was to the random points of the rainfall. For each cell with rainfall, the computed direction grid was called to determine which points will the water flow to. After determining the directions, the amount of rainfall found in the current cell was then distributed according to the amount precentage specified on the next node which was computed earlier during the flow trace. On this process, two of the flows are monitored, the catchment flow and the trace flow where the catchment flow records the final result of each of the cell in the DEM after the rainfall, whereas the trace flow is concerned on recording each of the amount of the nodes that was passed through during the computation. The traceflow mainly was computed for the image output of the program to determine the water flow.

E. Data Merging

After computing the catchment areas and the trace flow, the said local grids are then passed back to the master node where the master node's job is to collate the data and call the function to show the visual representation of the water flow trace. The MPI_receive function was not done in order of the nodes (ex.

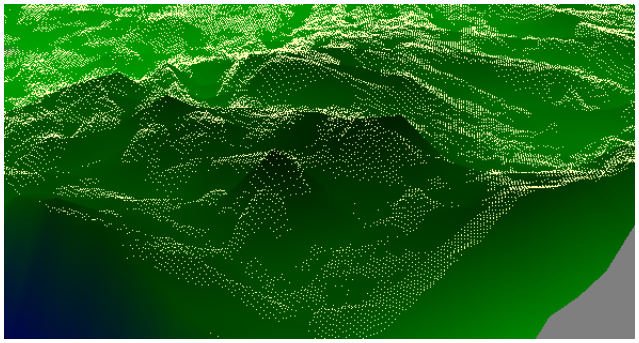
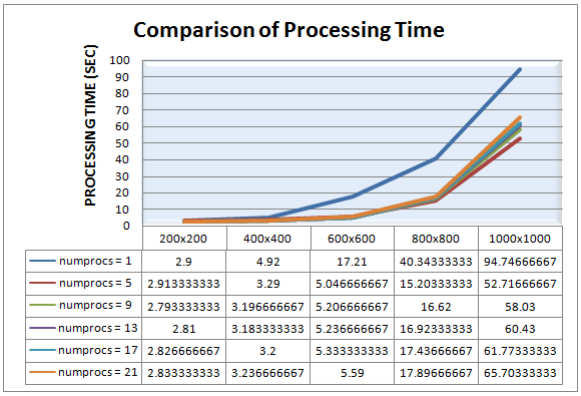
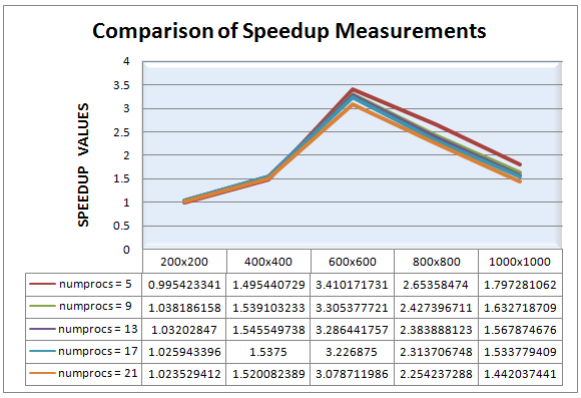


Fig. 4. 500x500 output image generated from 13.2552,123.686 coordinates



	200x200	400x400	600x600	800x800	1000x1000
numprocs = 1	2.9	4.92	17.21	40.34333333	94.74666667
numprocs = 5	2.913333333	3.29	5.046666667	15.20333333	52.71666667
numprocs = 9	2.793333333	3.196666667	5.206666667	16.62	58.03
numprocs = 13	2.81	3.183333333	5.236666667	16.92333333	60.43
numprocs = 17	2.826666667	3.2	5.333333333	17.43666667	61.77333333
numprocs = 21	2.833333333	3.236666667	5.59	17.89666667	65.70333333

Fig. 5. Comparison of the computation time given by sequential flowviz MD∞ and parallelflowviz.



	200x200	400x400	600x600	800x800	1000x1000
numprocs = 5	0.995423341	1.495440729	3.410171731	2.65358474	1.797281062
numprocs = 9	1.038186158	1.539103233	3.305377721	2.427396711	1.632718709
numprocs = 13	1.03202847	1.545549738	3.286441757	2.383888123	1.567874676
numprocs = 17	1.025943396	1.5375	3.226875	2.313706748	1.533779409
numprocs = 21	1.023529412	1.520082389	3.078711986	2.254237288	1.442037441

Fig. 6. Comparison of Speedup Measurements.

node 1 first then node 2), since there is no certainty that node1 will actually the first to finish in the local computation of the flow. The master node just waits if any of the nodes have finished computing and then notes the processor name of that node and collect all of the data needed from that node first before waiting for the result of the next node until all of the nodes have finished computing and the master node is done reconstructing the DEM grid.

F. Visual Representation

The 3D visual representation was rendered only by the master node using all the data passed to it computed by the slave nodes.

VI. Results and Discussion

A. 3D representation

The 3D visual representation of the terrain was rendered first followed by the flow trace which was rendered using GL_POINTS in shades of yellow (See Figure 3 on page 3).

The software was able to successfully implement the new triangular multiple flow direction algorithm on flowviz to calculate the catchment areas for the flow of water thus giving a more dispersed and realistic trace of the water flow from the initial straight downward direction flow.

The software also successfully applied data partition using the parallel programming paradigm and was able to compute for the catchment areas.

B. Performance Measurements

One of the simplest and most widely used indicators for a parallel program's performance is the observed speedup of a code which has been parallelized.

$$S_p = \frac{T_1}{T_p} \tag{1}$$

In equation(1), P is the number of processors, T_1 is the execution time of the sequential algorithm, and T_p is the execution time of the parallel algorithm with P processors. In order to compute for the speedup of Parallel Flowviz, the researcher created a sequential version of it which is basically just the original flowviz using the MD∞ algorithm. Then, the average time Parallel Flowviz and its sequential version took to execute was recorded for comparison.

Both programs were run on DEMs with dimensions 200x200, 400x400, 600x600, 800x800, and 1000x1000 as inputs and for the parallel algorithm, 4, 8, 12, 16 and 20 number of processors were used. Figure 5 shows the average elapsed time of computation for the different sizes of DEM run through different number of processors. Here, the graph shows that the parallel algorithm had a much faster elapsed time of computation than its sequential counterpart.

The ideal speedup, which is linear, is obtained when $S_p = P$. Figure 6 shows the speedup for the calculation of water flow directions using the MD∞ algorithm. There was an inverse relationship betwee the speedup results and the number of processor used. The speedup increased as the DEM data size approach the value 600x600 but it decreased when the DEM dimension size exceeded the 600x600 dimensions. This is caused by the increase in the cost of computation between the nodes as the DEM input size increases, to see the graphical representation of the comparison of the computation cost see figure 7.

$$Cost = T_p * P \tag{2}$$

The figure shows that the cost of communication is directly proportional to the number of the DEM data size and there

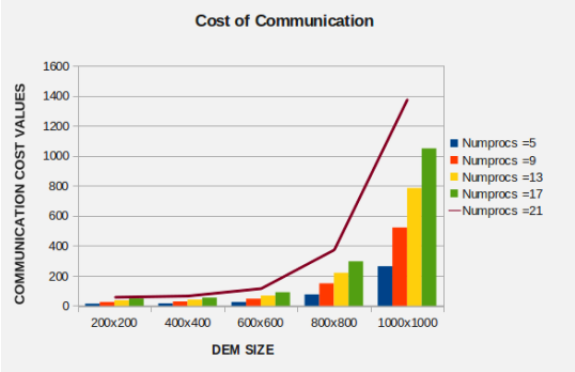
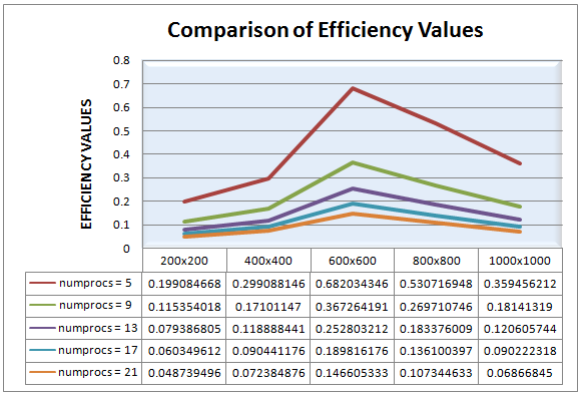


Fig. 7. Comparison of Communication Cost.



	200x200	400x400	600x600	800x800	1000x1000
numprocs = 5	0.199084668	0.299088146	0.682034346	0.530716948	0.359456212
numprocs = 9	0.115354018	0.17101147	0.367264191	0.269710746	0.18141319
numprocs = 13	0.079386805	0.118888441	0.252803212	0.183376009	0.120605744
numprocs = 17	0.060349612	0.090441176	0.189816176	0.136100397	0.090222318
numprocs = 21	0.048739496	0.072384876	0.146605333	0.107344633	0.06866845

Fig. 8. Comparison of Efficiency Values

was a notable increase in the cost values when the dimensions exceeded 600x600. Figure 5 shows the influence of the cost of communication on the processing time of the nodes when the DEM size reached past the 600x600 dimension.

$$E_p = \frac{S_p}{P} \tag{3}$$

In equation (3), the E_p refers to the efficiency of the parallel program, which is a value between zero and one, that estimates how well-utilized the processors are in solving the problem compared to how much effort is being wasted in communication and synchronization. Algorithms run on a single processor have an efficiency value of 1.

Figure 8 shows that the efficiency of this algorithm increased as the number of the DEM data size increased but only up to a certain dimension which is 600x600. Then, the cost of communication caused the decrease in efficiency of the algorithm past that data size. Also, the algorithm's efficiency is inversely proportional to the number of processors being used.

VII. Conclusion and Future Work

The results show that the implementation of parallel computing in Flowviz has improved the elapsed time of computation. However, it also shows that as the data input increases, there comes a point when the cost of communication becomes more costly than the flow computation so it is better to lessen the number of processors being used. The rendered 3D image also shows that the Triangular Multiple flow direction algorithm provided a more realistic visual representation and catchment computation for the trace of the water flow.

Future modifications of ParallelFlowviz should include into consideration partition of data not into grids but rather horizontally which would give the program more options when it comes to the number of processors that can be used and it would save time during the data partition and data merging because it will lessen the communication cost per node. A horizontal partition of data would allow an odd number of processors during the execution of the program whereas in this study, the data was immediately divided into half and each half was then further divided into another half or more depending on the specified number of processors which, as you can see, can only be multiples of four. It would also be an increase in performance if asynchronous communication will be used for the data transfer to save for the waiting time (MPI Isend and MPI Irecv).

Acknowledgment

Many thanks to the Advanced Science and Technology Institute (ASTI) High-Performance Computing (HPC) Facility and Philippine e-Science Grid (PSciGrid) for providing the cluster environment that was used in this study.

References

[1] J. A. C. Hermocilla and J. P. Pabico. (2003) Flowviz. [Online]. Available: http://flowviz.googlecode.com/
[2] E. X. T. Cheng, Ed., *Changes of flood control situation and adjustments of flood management strategies in China. Proceedings of the 2nd International Symposium on Flood Defence, Wu et al. (eds)*. Beijing, China: Science Press New York Ltd., 2002.
[3] K. Beven, "Rainfall-runoff modelling the primer." 2001.
[4] E. Z. J. Marsalek, E. W. Watt, and E. F. Sieker, Eds., *Flood issues in contemporary water management.* Malenovice, Czech Republic: Proceedings of a NATO Advanced Research Workshop on Coping with Flash Floods: Lessons Learned from Recent Experience, may 1999.
[5] G. E. Blelloch and B. M. Maggs. (2009) Flowviz. Carnegie Mellon University. [Online]. Available: http://www.buyya.com/cluster/v2chap1.pdf
[6] L. M. E. Silva and R. Buyya. (1999) Parallel programming models and paradigms. [Online]. Available: http://www.buyya.com/cluster/v2chap1.pdf
[7] J. P. Pabico. (2006) Parallel implementation of the d8 flow routing algorithm in a geographic information system. [Online]. Available: http://www.getcited.org/mbrz/11120313
[8] B. R. G. Maga, "Parallel streamline delineation,," 2007.
[9] R. Kiss, "Determination of drainage network on digital elevation models, utilities and limitations," *Journal of Hungarian geomathematics*.
[10] J. P. Wilson, "Terrain analysis tools for routing flow and calculating upslope contributing areas," 2002, presented at the Terrain Analysis for Water Resources Applications Symposium.
[11] J. Seibert and B. L. McGlynn, "A new triangular multiple flow direction algorithm for computing upslope areas from gridded digital elevation models, water resources," 2007, res., 43, W04501, doi: 10.1029/2006WR005128.
[12] K. e. a. Asanovic, "The landscape of parallel computing research: A view from berkeley.l," 2006.