On the Grid and Sensor Networks*

Vipul Hingne, Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Email: {vipul1,joshi}@cs.umbc.edu

Elias Houstis
Department of Computer Sciences & School of Engineering
Purdue University & University of Thessaly, GREECE
Email: enh@cs.purdue.edu

John Michopoulos
Computational Multiphysics Systems Lab
Naval Research Laboratory
john.michopoulos@nrl.navy.mil

Abstract

The pervasive computing environment and the wired Grid Infrastructure can be combined to make the Information Grid truly pervasive. Interesting applications can be built by utilizing the computational abilities of the Grid to answer queries on sensor data. In this paper, we identify some of the research issues and challenges in building an infrastructure to support such applications. We present the design and preliminary implementation of the proposed infrastructure, identify the tradeoffs relating to sensor accuracy consumption, and report experimental results from a sample scenario involving firefighting.

1. Introduction

The near future would see sensor networks being deployed widely for a variety of purposes that involve monitoring, from industrial process to the environment to physical perimeters and even homeland defense[1]. They will sense various attributes of the environment they reside in, as well as have the capability to perform computation and communicate with other sensors. Much of the existing efforts in sensor networks focus on applications where the raw data itself is relevant[17, 14], or at best relatively straightforward[19, 15] aggregates of it such as sum, average, max, etc. However, many practical applications based on data from sensor networks need much more complex data processing and analysis. We argue that the Grid provides the computing infrastructure which, when used in conjunction with sensor networks, can enable this new class of applications.

1.1. Motivation

Consider a building with temperature sensors embedded at various locations inside it. The sensors can communicate with some neighboring sensors, and with base-station(s) located in the building. How could such sensor networks be used in case of a fire. One obvious possibility is that firemen will carry small handheld devices which will be able to connect to the basestation using short range RF (802.11, Bluetooth) and query the sensors. The obvious query could be as simple as finding the temperature at a particular location, which would involve directly obtaining the a single sensed value. Another possibility would be to consider the average temperature in a room. Such queries have been the focus of much of the sensor data management efforts[17, 19, 15]. However, we can envision a scenario where neither the data, nor any simple aggregate of it is sufficient. A fireman might want to know how the fire will spread given the current sensed values of the temperatures and the diagram of the building. She might want to know if there will be a blowback if a particular door was opened to effect entry into the room, or whether a particular passageway will have acrid

* This work was supported in part by NSF awards ACI 0203958, 0205663, and IIS 9875433

Proceedings of the Fourth International Workshop on Grid Computing (GRID'03)
0-7695-2026-X/03 $ 17.00 © 2003 IEEE

smoke from the insulation smoldering. For such queries, the answer is derived from some complex computation performed on the sensed data values – in this particular instance a multi-physics simulation involving PDEs with the readings from various sensors as initial conditions. The approach needed to answer such queries is very different from the one that we would take to answer aggregation queries.

Clearly, the computation for solving such a query is beyond the capability of the sensor network, or even the basestation. An obvious solution, assuming that the basestation was connected to the Internet, would be to stream the data from the sensors to some remote computational resource on the grid. However, this consumes a lot of sensor energy, since transmission is a major energy depleter. Depending on a variety of factors (energy available, accuracy desired, sensor network configuration), alternate strategies are possible as well, as we detail in section 3.1.

1.2. System Overview

We have designed an infrastructure to allow combining the Grid with sensor networks which consists of the following primary components:

- Query Processor
- Decision Maker
- Grid Interface
- User Interface for Querying

In our initial implementation, we simulate the sensors for performing experiments to test the infrastructure. This is done using a simulator developed as a part of this project. The query processor parses and categorizes the queries submitted by users. The decision maker picks the computation models are to be used for a particular query, and how the computation should be partitioned between the sensors, the basestation, and the grid. The Grid will be used to perform computation which cannot be handled inside the sensor network or at the base station. Since the targeted user is a non expert, the user interface provide a selection type form to specify the queries and visualize the results.

We assume battery operated wireless autonomous sensors (such as the Berkeley Motes) which have local processing and communication capability on top of their sensing function. Conserving the energy of the sensors is one of our focus point in this work. In our experiments, we study the tradeoff between sensor energy consumed for different partitions of computation between the Grid and the sensors and the accuracy of the result for each partition.

2. Related Work

To the best of our knowledge, this is the first research effort which seeks to utilize the computational abilities of the grid to answer queries on streaming data from sensor networks. The work related to our effort is mostly in two different fields, the sensor database community and the ubiquitous grid community. There have been efforts in the sensor database community that look into efficient query processing in sensor networks and present various approaches for in-network processing in sensor networks. Attempts are being made to make the Grid ubiquitous by providing means of access to the Grid from mobile devices. We discuss these next.

Gehrke et al present a model for sensor databases in their Cougar Sensor database project[17]. In COUGAR, sensors are modeled using ADTs and the signal processing functions of the sensors are modeled by ADT functions which return the sensor data. More recently, in [19], they concentrate on in-network aggregation, interaction of in-network aggregation with the wireless routing protocol, and distributed query processing.

Franklin et al have proposed a query plan data structure called Fjords(Framework in Java for Operators on Remote Data Streams)[14] to allow queries combining push-based sensor sources and conventional pull-based sources. Madden et. al propose aggregation of data in sensor networks as part of their TinyDB project TAG (Tiny Aggregation)[15]. In TAG, Madden et. al show that performing the computation for certain type of aggregate queries inside the sensor network result in saving the energy of the sensors and thus lengthen the lifetime of the sensor network. In [11], they extend the TinyDB sensornet query engine to support more complex data analyses, like topographic mapping, wavelet based compression, and vehicle tracking.

Projects such as Globus[6] and Legion[10], and the proposed OpenGrid environment all allow for computation distribution, enabling high-performance applications to exploit diverse, geographically distributed resources. Accessing the grid/sensor infrastructure from wireless thin clients is clearly a needed component of our system. This necessitates the notion of not just distribution, but the partition of computation across asymmetric thin client-fat server systems[5, 16]. Work on wireless access to the grid infrastructure using proxy based approaches to split the computation and transform return content was done by the SciencePad project[4, 5] in the late 90s. More recently, the CAROUSEL [7, 8] Community Grid for PDA project is using a similar approach to develop environments to support ubiquitous access to Community Grid systems from various small wireless devices like Personal Digital Assistants(PDAs). The iMASH project [18], involves the development and deployment of a network system that supports anytime, anywhere, on any platform access to the electronic patient records database for healthcare providers. Again, the basic idea is related to content transformation using proxying.

3. System Design

There has been a lot of work in the distribution of (sub)computations of a given task in the Grid computing community, as well as computation partition for wireless access. However, a slew of new issues arise when the sources of data are tiny energy and resource constrained sensors, and the network connectivity is ad-hoc and the topology dynamic. In this section, we present our approach to dynamically partitioning computation between the Grid and the pervasive elements like sensors. We also discuss the implementation of our infrastructure which allows for combing the grid with sensor networks

3.1. Factors Influencing Computation Partition

The decision to partition the computation is based on several factors. The first and most essential factor is the amount of computation required for generating the answer to a particular query. Ancillary issues along these lines involve knowing whether the sensors are programmable on the fly, and if so, whether the particular computation desired can be accommodated by their limited physical memory, CPU speed and battery life. Another important parameter is the amount of data transfer required for evaluation of the query. In the current generation sensors, a single transmission is almost 3 orders of magnitude more energy consuming than a single instruction execution. A related issue is where the transmission is directed. If a sensor is communicating with its neighbor, it probably uses much less energy than if it is communicating directly with the basestation further away. As is well known, to a first order of approximation the energy consumed in transmitting is directly proportional to the square of the distance over which the transmission occurs. The amount of data transfer for a simple query like finding the temperature at a given sensor is less than that of an aggregation query for instance. Complex queries that require all data to be sent to the grid will clearly demand even more energy, as will any query which is continuous in nature , like finding the temperature at a particular sensor every 10 seconds. In sensor networks, preserving the energy of the sensors is of prime importance. So estimates of energy consumption of sensors to evaluate a query with each of the above approach are desirable.

The desired response time can be another crucial factor, especially for near real-time queries, the turn around time is crucial. Network size and topology is also important. All networks may not be of the same size, so the number of sensors in the network would wary. Different networks would have different network topology. The data routing technique used in the network would not be the same for all networks. A particular network may use flooding technique to route data, while another may use gossiping. Different sensors may generate data with different rates.

We investigate three different solution models that can be used to gather data and perform the computation required to answer a query. In a simple model, all sensors would send their data to the base station. The base station would then perform the computation over the data. Cluster based models can enable the computation to be carried out in the sensor network. Sensors are divided into clusters and each cluster has a cluster head. Cluster heads aggregate information from the sensors in individual clusters, computes the needed data, and sends it to the base station. Another way to perform in-network aggregation is to use aggregation trees[15]. Data centric routing techniques can be used to form aggregation trees in sensor networks. Data would be routed and aggregated through the aggregation trees. Most importantly, the grid can be used to perform the computation. The data would be transferred to the grid through the base station. The computation would be done in the grid and results would be returned to the base station. For a given query, one or a combination of the models above can be used to perform the computation.

3.2. System Components

Our system comprises of three major components: Query Processor, Decision Maker and the Grid Interface. Query processor analyzes the query and categorizes it into one of the types mentioned above. Decision maker decides which model to use by looking up the data-table generated a-priori. The simulator simulates the solution model for the query and return the results. The grid does the complex calculations required to answer some of the complex queries.

3.2.1. Query Processing We assume that the users specifies the information needed in some declarative, SQL-like format. In particular, we modify the format proposed by Madden et. al[15] in two significant ways to become:

```
SELECT {func(), attrs} FROM sensors
WHERE { selPreds }
COST { cost limitation }
EPOCH DURATION i
```

First, we allow for any arbitrary function of the sensor data to be specified in the SELECT clause of the query. The only requirement is that we have the code to evaluate the function and the components needed to compute the function have to be discoverable from the basestation. Discovering appropriate computation components is an active research topic in the grid community, and simple syntax based approaches are a part of most Grid systems. We have proposed more complex semantic driven matching techniques that are described elsewhere[13, 3, 2]. This feature will al-

Proceedings of the Fourth International Workshop on Grid Computing (GRID'03)
0-7695-2026-X/03 $ 17.00 © 2003 IEEE

low us to address a wide range of application queries over the sensor network.

Second, we introduce the COST clause. The COST clause can be used to provide constraints on the query which aren't directly based on the data on which the query is performed, and hence not a part of the WHERE clause. For example, the COST clause can enable the user to have some control over the sensor energy consumed while answering his query. At present, we allow the COST to be specified in terms of the sensor energy or the accuracy of the result. If the query is critical, the user can specify that the system use as much energy as is required to get the result as soon as it can. If the query is not so critical, then the user can specify how critical the query is by suggest either moderate or low energy consumption for the query.

A point to note here is that not all clauses have to be specified in each query. Only the SELECT clause and the FROM clause are the required clauses. Other clauses are optional.

3.2.2. Decision Making Whenever a query is submitted, decisions need to be made as to where and what of the computation, as well as the configuration of the underlying sensor network.. This issue is similar to the "Algorithm Selection" problem in the PSEs. Our approach to make these decisions is based on data collected using simulations (and eventually, from real systems). Such an approach has been shown to be very effective (e.g.[12]) for algorithm selection. We have collected data regarding the amount of computation, data transfer, energy consumption, accuracy of result for all the query types supported by our system for different configurations of the sensor network.

The computation model is picked based on the query type, cost requirements, known features of the network at hand. A look-up is then done on the data collected in prior simulations. The computation model which gave the best results and has the closest values of the these parameters is selected. Eventually, much like the Pythia system[12], we hope to be able to use machine learning approaches to build implicit models of the data.

The decision maker also decides whether to reconfigure the network in terms of the size of the clusters into which the sensors are grouped by increasing the neighbors they can talk to. This decision is made purely to make the network more energy efficient or make the results more accurate. For example, if the current neighbor connectivity is 2 and a continuous query is issued which needs the average temperature to be reported every 10 seconds. In this case, if the neighbor connectivity is increased to 10 and the network reconfigured, lot of sensor energy can be saved. (Our results, explained in detail in the next section, show that almost 50% of energy can be saved by increasing the neighbor connectivity for aggregate queries.) Similarly, if the neighbor connectivity is 10 and a query is issued which requires the temperature distribution with high accuracy, then the neighbor connectivity is decreased and the network reconfigured to obtain higher accuracy.

As mentioned, at present the decision maker uses fairly simple lookup type mechanisms. However, our system is designed to be able to plug in any decision making algorithm that we (or others) build in future.

3.2.3. The Grid Interface The system also needs to know how to interact with the Grid, for which the Grid interface module is used. In our preliminary implementation, we simply connect to a remote machine that runs FreeFEM+ [9] to solve these PDEs. In our implementation, the initial conditions for the PDE to be solved come from the data received from the sensor network. We then interpolate the data from the sensor network to generate a function which provides the initial conditions for the PDE. The next step is to put the initial conditions in the *.edp* file for the PDE and give it as input to freeFEM+. The result from freeFEM+ is a set of graphic files, which we stream to the user. In ongoing work, we are adding modules to the system to be able to interact with the Semantic Grid / Open Grid based architectures.

3.3. Sensor Simulator

It is possible to hook up actual sensors to sense and stream data to be used along with our system. However, we decided to simulate the sensors for our experiments to give us more flexibility in building a system which can work not only with the current generation of sensors but also with other sensors which may come up in the future. Unfortunately, there are very few sensor simulators that are publicly available, and most of them focus on the network layer. These simulators provide limited flexibility for application level development.

The sensor simulator we have built can simulate a wide variety of sensor networks. All the parameters of sensor network to be simulated can be specified in a configuration file. In particular, the following parameters are configurable

- The extent of the sensor network in 3 dimensions
- Number of sensors in the network
- The location of each sensor can be either read in from a file or the sensors can be distributed within the given extent uniformly or randomly
- The number of neighboring sensors that each sensor can talk
- The communication radius of each sensor
- The location of the basestation
- The topology of the sensor network. We allow two different topologies, cluster network and aggregation trees

- The Energy model of the sensor. Sensors can be based on the radio model of energy consumption or they can specify the ratio of communication energy to computation energy
- The kind of sensing tasks that the sensor is capable of

The simulator is written in Java and utilizes threads to simulate each sensor. After reading the configuration file, the network is initialized with the parameters specified in the file. A query can be submitted to the sensor network once it is initialized. The base-station propagates the query to all the sensors required for answering the query. Each sensor performs the sensing task required to answer the query. Once it senses the environment, the sensor forwards its reading either to another neighboring sensor or to the base-station. The network topology is essentially a multi-hierarchy which establishes how sensors forward their readings to the base station. Note that a hierarchy is required for two main reasons. Firstly, we do not assume that all the sensors have the capability to transmit readings to the base station and hence they have to transmit it via other sensors. Secondly, in the radio model of energy consumption, which is a standard communication equipment for many commercially available intelligent sensors, the energy consumed per transmission is proportional to the square of the distance over which the transmission is made, so multiple short hops of communication result in saving energy over single long hop.

4. Experiments and Results

4.1. Infrastructure Validation

4.1.1. Sensor Network Simulator The simulator was tested on pentium celeron 900MHz CPU with 256MB RAM. The simulator was thoroughly tested with all possible variations of the configurable parameters of the simulator. Each configurable parameter was independently varied and the desired change was observed in the simulator and was found to be consistent with the expected result.

The scalability was tested on a log scale. The simulator was found to be running with expected results for up to 10,000 sensor nodes. For small configurations, parameters such as energy drained or messages exchanged were hand calculated and verified. At about 100,000 nodes the simulator ran out of memory and could not complete the simulations. We anticipate that on newer 64 bit machines, we should be able to simulate another 2-3 orders of magnitude in terms of the number of sensors.

4.1.2. Query Processor and Decision Maker The query processor was thoroughly tested on the simulator and was found to be running in desired fashion. All possible query types were tested on the simulator and the queries were processed and classified as expected.

The decision maker used the data generated by conducting extensive simulations for making the decisions. Given the parameters of the network at hand and the query submitted to the query processor, the decision maker was able to look up the right values in the data table in order to make the decisions regarding the computation partition and other things like network reconfiguration.

4.2. Scenario for the Experiments

In our experimental scenario there is a 10m * 7m room in a building that catches fire. It is assumed that there is a single source of fire in the room. There are a hundred temperature sensors in the room which can sense the temperature and report the readings back to a base-station. The base-station is capable of communicating with the Grid. Firemen want to initiate relief operations to control the fire. They have mobile devices which can communicate with the base-station to query the sensor network in the room. In particular, the firemen want to know the effect of opening the door of the room which has caught fire.

To answer such a query, we set up and solved a convection-conduction PDE with the temperature readings from sensors as initial conditions . The PDE for the case of conduction is

$$\frac{\partial T}{\partial t} - \rho c_p \nabla^2 T + aT = f$$

For convection, the PDE is

$$\frac{\partial T}{\partial t} + e\nabla T + aT = f$$

The terms aT and f are source terms and the term $e\nabla T$ is the convective term. c_p is the heat capacity and ρ is the density of the medium. Clearly, this makes simplifying assumptions such as a point source of fire, and ignores the more complex methods of heat propagation, but serves as a good first order approximation.

We use Freefem+ [9] to solve these PDEs. In Freefem+, a mesh is defined on the geometry of the room. Temperature readings at the mesh-points are used as initial conditions to solve the desired PDE. We do not assume that sensors are placed at all the mesh-points. Instead we use the sensor readings to interpolate the temperature values at the mesh-points. A certain degree of error is introduced due to the interpolation.

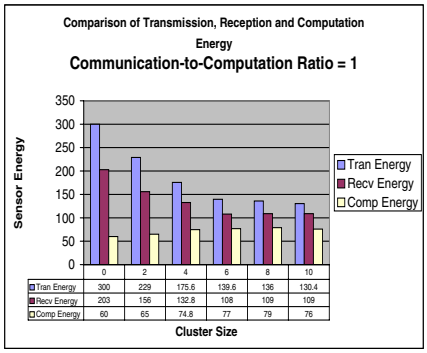
4.3. Studying Tradeoffs

A sensor typically consumes energy for three purposes. Firstly, sensors require energy to transmit data. Secondly

they consume energy when they receive data. Finally, sensors consume energy when they perform some kind of computation. The energy consumed while transmitting or receiving data is directly proportional to the size of the data, i.e. the number of bits required to represent the data. The energy consumed while performing computation is proportional to the length of the computation, i.e. the number of instructions executed by the microprocessor in the sensor.

Instead of reporting all the sensor readings to the base station, if we perform some kind of in-network aggregation and report only the partially aggregated values to the base-station we can save some energy in transmission and reception. Here, we would spend more energy in the form of computation, so one of the motives is to study the communication-computation energy tradeoff.

One of the parameters in studying the communication-computation energy tradeoff is the communication energy to computation energy ratio, i.e. the amount of energy consumed to transmit one bit of information as compared to the amount of energy consumed while executing one instruction. Most of the currently available sensors, like Berkeley motes, use radio model of communication, where the communication-to-computation ratio is about 800. We, however, do not believe that this would be necessarily true of next-generation sensors that use passive communications (smart dust, RFID based). So while carrying out our experiments we have used 3 different kinds of communication-to-computation ratios, i.e. 1, 10 and 800 (for radio model of communication).

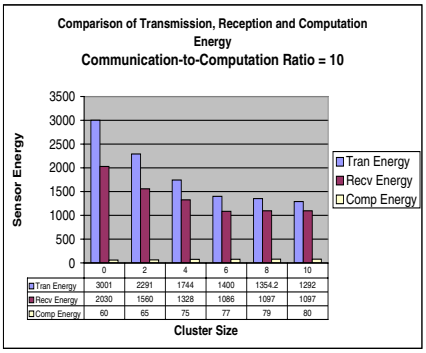


	0	2	4	6	8	10
Tran Energy	300	229	175.6	139.6	136	130.4
Recv Energy	203	156	132.8	108	109	109
Comp Energy	60	65	74.8	77	79	76

Figure 1. Energy Consumption Ratio for Communication to Computation Ratio = 1

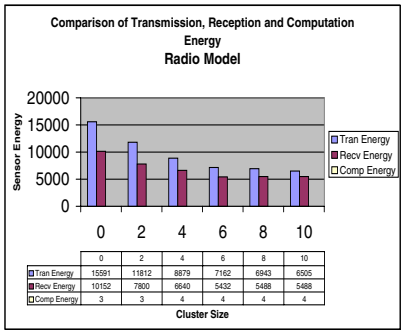
4.4. Cluster Network

We use a multilevel-cluster network of sensors for data propagation. A single cluster can consist of 2,4,6,8 or 10



	0	2	4	6	8	10
Tran Energy	3001	2291	1744	1400	1354.2	1292
Recv Energy	2030	1560	1328	1086	1097	1097
Comp Energy	60	65	75	77	79	80

Figure 2. Energy Consumption Ratio for Communication to Computation Ratio = 10



	0	2	4	6	8	10
Tran Energy	15591	11812	8879	7162	6943	6505
Recv Energy	10152	7800	6640	5432	5488	5488
Comp Energy	3	3	4	4	4	4

Figure 3. Energy Consumption Ratio for Radio Model of Energy Consumption

sensors depending on simulation parameters. Each cluster has a cluster head. All sensors send their data to their cluster-head. On receiving data from all sensors in its cluster, the cluster-head sends all the data to the next-level cluster head. To reduce the energy consumption, first level cluster heads can aggregate the readings of the sensors in its cluster and send only a single reading instead of multiple readings. The number of levels in the cluster network is

$$\log_s n + 1$$

where s = size of cluster and n = number of sensors

4.5. Accuracy

Each sensor reading consists of the sensor location and the sensed temperature value. When aggregating more than one reading, the average temperature value and average location is reported. This results in loss of accuracy while interpolating the values for mesh-points. The accuracy metric measured in the experiment is calculated as follows:

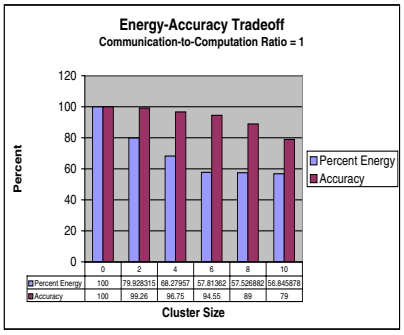


Figure 4. Energy-Accuracy Tradeoff for Communication to Computation Ratio = 1

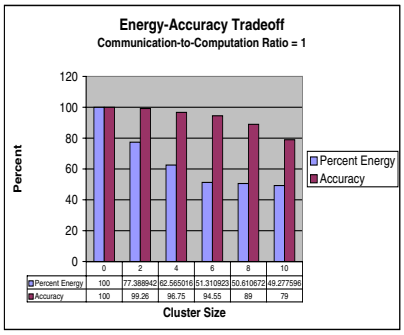


Figure 5. Energy-Accuracy Tradeoff for Communication to Computation Ratio = 10

The temperatures at all the mesh-points are measured without any interpolation. The PDE is then solved with these boundary conditions and the final values at each mesh-point are noted. Now, to find the accuracy of a given cluster-configuration and aggregation scheme, the temperature values at the mesh-points are calculated by interpolating the readings reported with the scheme. The PDE is then solved with these new boundary conditions and the final values at each mesh-point are again noted. The percentage error at each mesh-point is then calculated. The average of percentage error at each mesh-point is subtracted from 100 to give the accuracy.

4.6. Experiments

The experiments were carried out for 3 different communication to computation ratios, i.e. 1, 10 and 800(for radio model), because we wanted to study how much does the energy spent in computation contribute to total energy consumption of the sensor. The query was to find the effect of

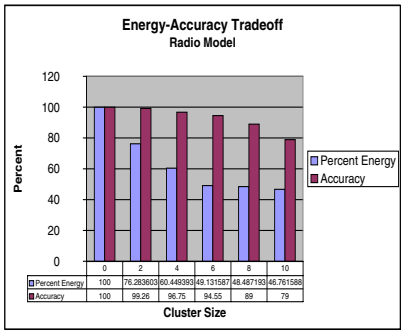


Figure 6. Energy-Accuracy Tradeoff for Radio Model of Energy Consumption

opening the door of the room which is on fire. For each ratio, the cluster size of sensor network was varied from 2 to 10 in increments of 2. The average transmission energy, reception energy, computation energy, total energy consumed and accuracy of the result were measured. Also, the experiment was repeated while performing no in-network aggregation and the readings reported as cluster size 0 correspond to this experiment. One set of graphs show the comparison of transmission energy, reception energy and computation energy for different cluster size. The other set of graphs show the Sensor Energy -Accuracy tradeoff.

4.7. Explanation of Results

We observe in figures 1, 2, 3 that the computation energy increases while the communication energy decreases as the cluster size increases. The amount of computation would increase with the increase in the size of the cluster, so the increase in the computation energy is justified. Also, as the cluster size increases, the number of results sent to the base station, which is equal to the number of clusters, decreases, hence the reduction in the communication energy. Also we note that for higher communication to computation ratio, the cost of computation is negligible as compared to the cost of communication. However, for lower ratios, the cost of computation is quite significant.

The second set of graphs, figures 4, 5, 6 provides more interesting results. We notice that we can get roughly 90% accurate results while spending roughly 50% of max energy consumption.

Another interesting observation is that the energy consumed decreases less dramatically as we attempt to increase the cluster size. This is because it gets increasingly difficult to find more nodes within the communication range of the sensor. So, even if we allow the maximum cluster size to be say 10, many of the clusters have less than 10 sensors because there may not be enough other sensors within a sen-

sor's communication range. Increasing the communication range of a sensor would further increase the communication energy and would lead to less accurate results, so is not attempted.

To summarize the most important result, significant reduction in sensor energy consumption is possible with slight decrease in accuracy, by doing some computation in the sensor network and sending less data outside the sensor network.

5. Conclusion and Future Work

We have designed and created the preliminary prototype of the infrastructure to utilize the Grid to answer queries on data from sensor network, We also present an application to fire hazard mitigation. The results we obtained show that for our chosen application, significant reduction in sensor energy can be achieved by slightly relaxing the constraints on the accuracy of the results and clever partitioning of computation between the sensors and the Grid. Energy consumed for performing computation in sensors can be significant in case the communication to computation ratio is less. We expect that such results can be obtained for many real life applications.

In future work, we would like to collect more data on energy consumption, response time and accuracy of results, and then apply machine learning techniques to select the right way to partition the computation. We also want to continue looking at more complex fire scenarios, with multiple sources of fire, smoke propagation, and degradation of structures, etc. Finally, we are working to integrate the system with a real grid system.

References

[1] S. Avancha, J. Undercoffer, A. Joshi, and J. Pinkston. Secure sensor networks for perimeter protection. *Special Issue of Computer Networks on Wireless Sensor Networks*, 2003.

[2] D. Chakraborty, F. Perich, S. Avancha, and A. Joshi. Dreggie: A smart service discovery technique for e-commerce applications. In *Proc. Workshop on Reliable and Secure Applications in Mobile Environments, 20th Symposium on Reliable Distributed Systems*, October 2001.

[3] H. Chen, A. Joshi, and T. Finin. Dynamic sercice discovery for mobile computing: Intelligent agents meet jini in the aether. *Baltzer Science Journal on Cluster Computing*, 4(4), Oct 2001.

[4] T. Drashansky, A. Joshi, S. Weerawarana, and E. Houstis. Sciencepad: Scientific computing in a ubiquitous environment. *Intl. J. Microcomputer Applications*, 15(3), 1996.

[5] T. Drashansky, S. Weerawarana, A. Joshi, R. Weerasinghe, and E. Houstis. Software architecture of ubiquitous scientific computing environments. *ACM-Baltze Journal on Mobile Networks and Applications*, 1, 1997.

[6] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[7] G. Fox, H. Bulut, K. Kim, S.-H. Ko, S. Lee, S. Oh, S. Pallickara, X. Qiu, A. Uyar, M. Wang, and W. Wu. Collaborative web services and peer-to-peer grids. In *Collaborative Technologies Symposium (CTS'03)*, 2003.

[8] G. Fox, D. Gannon, S.-H. Ko, S. Lee, S. Pallickara, M. Pierce, X. Qiu, X. Rao, A. Uyar, M. Wang, and W. Wu. *Peer-to-Peer Grids*. 2002.

[9] freeFEM. World Wide Web, `http://www.freefem.org`.

[10] A. Grimshaw and W. Wulf. The legion vision of a worldwide virtual computer. *Comm. ACM*, 40(1):39–45, 1997.

[11] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Towards sophisticated sensing with queries. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, Palo Alto, March 2003.

[12] E. N. Houstis, A. C. Catlin, J. R. Rice, V. S. Verykios, N. Ramakrishnan, and C. E. Houstis. PYTHIA-II: a knowledge/database system for managing performance data and recommending scientific software. *ACM Trans. Mathematical Software*, 26(2):227–253, 2000.

[13] A. Joshi and L. Xu. A jini based framework for a component recommender system. In *Proc. Sixteenth IMACS World Congress*, August 2000.

[14] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *ICDE*, 2002.

[15] S. Madden, M. J. Franklin, J. M.Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *5th Annual Symposium on Operating Systems Design and Implementation(OSDI)*, Boston, December 2002.

[16] S. Markus, S. Weerawarana, E. Houstis, and J. Rice. Scientific computing via the web: The netpellpack pse server. *IEEE Comp. Sci. Engr*, 4:43–51, 1997.

[17] P.Bonnet, J.Gehrke, and P.Seshadri. Towards sensor database systems. In *Conference on Mobile Data Management*, 2001.

[18] T. Phan, G. Zorpas, and R. Bagrodia. An extensible and scalable content adaptation pipeline architecture to support heterogeneous clients. In *22nd International Conference on Distributed Computing Systems*, July 2002.

[19] Y. Yao and J. Gehrke. Query processing for sensor networks. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, January 2003.

Proceedings of the Fourth International Workshop on Grid Computing (GRID'03)
0-7695-2026-X/03 $ 17.00 © 2003 IEEE