

A Hardware and Software Computational Platform for HiPerDNO (High Performance Distribution Network Operation) Project

Stefano Salvini
Oxford e-Research Centre,
University of Oxford
Oxford, UK
stef.salvini@oerc.ox.ac.uk

Piotr Lopatka
Oxford e-Research Centre,
University of Oxford
Oxford, UK
piotr.lopatka@oerc.ox.ac.uk

David Wallom
Oxford e-Research Centre,
University of Oxford
Oxford, UK
david.wallom@oerc.ox.ac.uk

ABSTRACT

The HiPerDNO project aims to develop new applications to enhance the operational capabilities of Distribution Network Operators (DNO). Their delivery requires an advanced computational strategy. This paper describes a High Performance Computing (HPC) platform developed for these applications whilst also being flexible enough to accommodate new ones emerging from the gradual introduction of smart metering in the Low Voltage (LV) networks (AMI: Advanced Metering Infrastructures).

Security and reliability requirements for both data and computations are very stringent. Our proposed architecture would allow the deployment of computations and data access as services, thus achieving independence on the actual hardware and software technologies deployed, and hardening against malicious as well as accidental corruptions. Cost containment and reliance on proven technologies are also of paramount importance to DNOs.

We suggest an architecture that fulfills these needs, which includes the following components for the HPC and Data Storage systems:

- Hadoop Distributed File System
- a federation of loosely coupled computational clusters
- the PELICAN computational application framework

Categories and Subject Descriptors

J.2 "Computer Applications in Physical Sciences and Engineering"

General Terms

Management, Performance, Design, Reliability, Security.

Keywords

Smart Grid, High Performance Computing Applications, Systems design

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. HiPCNA-PG'11, November 13, 2011, Seattle, Washington, USA. Copyright 2011 ACM 978-1-4503-1061-1/11/11...\$10.00.

1. INTRODUCTION

A lower carbon future means that millions of homes and businesses will have smart meters; that there will be an increasing use of small- and medium-sized solar, wind, and combined heat and power production systems embedded in the electricity network; that in the longer term there may be millions of electric vehicles needing to charge. Monitoring and controlling these so-called 'active electricity networks', will require new sensors and instrumentations and will generate vast amounts of data needing near-to-real-time analysis. Obviously, any technologies being developed now, would need to take into account these considerations.

The HiPerDNO project is developing new applications to enhance the operational capabilities of Distribution Network Operators (DNO) and to support their exploitation of the new data available to ensure that networks are in future able to perform much nearer to their operational capabilities. These will necessarily require an advanced computational strategy for their delivery. Example functions include Distribution State Estimation (DSE), Condition Monitoring (CM), and automated power restoration, described below. It is expected that Data Mining (DM) technologies could provide major benefits, for example better equipment failure prediction, more tightly defined RCM (Reliability Centered Maintenance) strategies, etc.

A common, low cost, High Performance Computing (HPC) platform is required for this wide range of current and future tasks, capable not just of handling Medium Voltage (MV) networks and their needs, but flexible enough to accommodate the new range of applications needed by the gradual introduction of smart metering in the Low Voltage (LV) networks, allowing for the introduction of different Advanced Metering Infrastructures (AMI) strategies. These applications and data requirements impose high demands on the IT infrastructure, on its computational power, storage capacity, communication bandwidth, scalability and security.

It should be noticed that advanced computational infrastructures are already deployed for Transmission Networks, which are well instrumented with sensor devices. The use of advanced computing for Distribution Networks is novel: networks are topologically much more extensive and complex and only a fraction of the network components are provided with sensors, etc. DNOs are as yet reluctant to invest in major changes to their practice, to their networks (by deploying large number of sensors, thus incurring in considerable costs) and in new, as yet unproven,

possibly disruptive technologies. Low cost and reliance on proven technologies are then of paramount importance.

We assume that the Distribution Management System (DMS) is at the core of the DNO activities. The DMS receives data from the network and elsewhere, provides data storage and retrieval, performs data analysis and computational resources and allows operators to interact with the network. We would like to stress that this is a conceptual description that reflects functionally current practice. Details of implementation and terminology may vary across DNOs.

This paper does not concentrate on either the DMS or on data acquisition from the network but concentrates on the **services** required by the DMS to access and store data and to carry out the required computations.

Viewing both data access and computation as services is at the very core of our proposal. Requests are issued by the DMS: these can be manual, originating from an operator, automatic, such as in the case of continuous network monitoring, or triggered by events, such as network component failure.

In this report we present a low-cost architecture that fulfills these needs. In the first section we briefly describe the overall application space with particular emphasis on the aspects that have a most direct impact on the HPC system. The next section analyses security risk factors and mitigations for the HPC system. The following sections examine in detail the components of the solution we propose, the Data System and the HPC Engine. This is followed by an analysis of the system size, based on data provided by project partners. This will be finally followed by our conclusions.

2. DATA AND APPLICATIONS CHARACTERISATION

Data flows into the system at varying rates and from different geographical locations. This can be visualised as a “data cube” (see Figure 1), where

- The “horizontal planes” correspond to data across the network at particular times. These planes may well be incomplete, as different measurement devices could operate asynchronously.
- The “vertical columns” include the time series for the corresponding small patches of the horizontal planes.

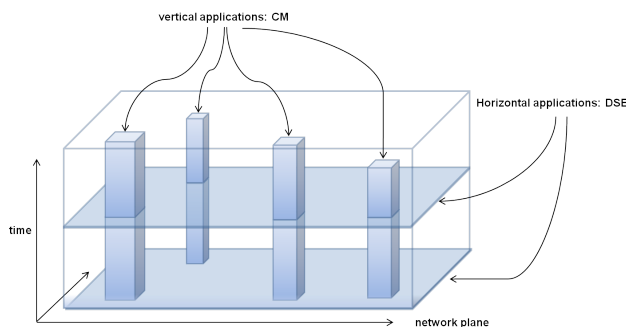


Figure 1: A schematic representation of the data space

This representation can help in analysing the applications requirements. Current applications could be classified as

- “Horizontal” problems. Current approaches to DSE; Power Restoration Algorithms (PRA).

- “Vertical” problems. CM; single device DM schemes.

It is likely that future applications will straddle this classification, e.g. DSE requiring historical analysis of data; DM across historical DSE data and specific collections of measurements; etc.

Horizontal problems will require enhanced computational power and consist of closely coupled components (however, see below for DSE considerations). Thus, they would require HPC techniques (including, of course, parallelism) inside the application itself. A parallel computational platform would be necessary.

Most vertical problems would require less concentrated computational power per problem, but many more problems would have to be run as concurrent, independent applications. The same platform could then be required to run a large number of serial (or very modestly parallel), independent jobs.

Both sets of applications could be run at the same time and the HPC platform must be able to cater for both.

2.1 Distribution State Estimation (DSE)

While State Estimation algorithms are commonly used in transmission networks, their use on MV (Medium Voltage) distribution networks is novel[1]. Currently, many of the nodes in an MV network are not instrumented and, due to the costs involved in adding metering devices, the situation is unlikely to change in the near future.

Given a set of measurements (real or virtual), the network topology and the characteristics of the devices, DSE solves an optimisation problem which provides an estimate of the current state of the network. A typical MV network would include many devices (thousands or more), thus requiring highly efficient and parallelized algorithms. Traditional optimisation methods,[2,3] such as WLS (Weighted Least Squares) and WEM (Weighted Error Modulus) are based on Newtonian approaches and vary on the characteristics of the metrics employed. Unfortunately, such methods have historically proved difficult to parallelise.

Alternatively, genetic and stochastic algorithms have been suggested, such as the Differential Evolutionary Algorithms (DEA) and Particle Swarm Optimisation (PSO) investigated within other parts of the HiPerDNO project [4]. They are easier to parallelise but their scaling with network size raises concerns about their suitability for large networks.

The networks studied have radial topology, hence splitting the networks into branches has been considered in two different ways. The first would deploy a number of measurement devices at critical positions in the network: branches would be fully disjoint (“zonal” approach) and each branch could be solved independently. The second method would solve each branch independently including “boundary” nodes shared with other branches, then matching the boundary nodes iteratively. Matching procedures are not trivial and convergence difficulties may ensue. The zonal approach involves extra costs in deploying measurement devices: this is a matter of policy for the DNOs, beyond this paper.

2.2 Condition Monitoring (CM)

CM applications analyse the time series of measurements on network components (underground cables, transformers, etc) to detect impending failures or maintenance requirements [5] and have a small footprint on the “horizontal” plane. In future, Data

Mining techniques (see below) will play a major role in detecting patterns and situations leading to major events to be flagged through the CM. Several devices can be analysed independently and asynchronously (“embarrassingly” parallel across devices). In future, it may be envisaged that information on the overall network, such as from the DSE, could be used in conjunction with detailed device measurements, thus requiring far more complex solutions.

2.3 Data mining (DM)

Data Mining will become one of the core activities of a DMS and this is reflected in its high profile within HiPerDNO. DM technologies and techniques [6] could prove invaluable in analysing device sensor data for CM. Currently, it is expected that most Data Mining will lie at the “vertical” end of the applications spectrum, thus offering the potential of analysing sensor information from several devices in parallel, non-communicating processes. There appears to be little scope for parallelism within DM applications, but new algorithms may well change this and the HPC Engine must allow for this.

2.4 Advanced Metering Infrastructure (AMI)

In future, AMI capabilities will play an important role for DNOs. Although no formal specifications yet exist, it is possible to surmise that AMIs will result in greater required bandwidth from the network measurement devices to data storage, subject to the desired level of data aggregation, and increasing computational power to analyse the data[7]. AMI information will also be used to define the boundary conditions for DSE approaches. The HPC system must be capable to tackle these as yet un-quantified challenges without requiring an overall redesign.

3. SYSTEM REQUIREMENTS

We give an overall view of the HPC system envisaged in Figure 2. The system has two levels, an *Operations* and an *HPC Level* with the Distribution Management System (DMS) as the only gateway between them. Network data are conveyed to the DMS and then passed through to the HPC level. The HPC Level consists of two separate components:

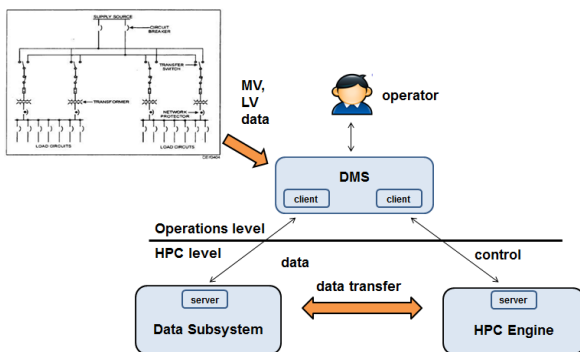


Figure 2: System Overview

- **The HPC Data System (HPCDS).** All data are stored on the HPCDS, and can be accessed and stored on it by the HPC Engine and the DMS.
- **The HPC Engine (HPCE).** This carries out all computations requiring HPC. The DMS requests computations to be carried out, manually or automatically. The HPCE then requests data from the HPCDS, notifying on completion the DMS while

storing the results on the HPCDS. The DMS can then issue a request to the HPCDS for the results.

This design followed a number of guidelines:

- **Separation of functions:** data can only be stored/accessed on the Data System; computation control and monitoring can only connect from the DMS to the HPC Engine. Neither can data be accessed nor computations carried out except through the DMS.
- **Client-server model.** The DMS clients would require either data or computation from appropriate servers residing within the HPCDS and HPCE. This has two benefits:
 - **Isolating the DMS** from the HPCDS and HPCE details. New technologies, upgrades etc. can be more easily deployed. A range of different DMSs are employed by DNOs: our suggested client-server model minimises the modifications required by different DMSs to access the HPC mechanisms proposed here.
 - **Decoupling of the back-end technologies** from the front-end, thus allowing relatively independent upgrading of the two.

The transparency of the HPCDS and HPCE also allows easier fulfillment of security and resilience requirements. Some specific points deserve careful consideration;

1) Resilience

DNO Operation is a mission-critical system, irrespective of the state of the network and of the data and computational systems. Hardware and software failures can be expected to occur, therefore the HPC system and applications must be able to handle them to prevent data loss and allow recovery mechanisms.

2) Accessibility

The Data Systems must allow adequate throughput, of course: both concurrent and parallel I/O technologies would be required. Data could be accessed by different applications with identification of data through the addition of metadata.

3) Performance

Performance is of paramount importance with applications such as DSE and Network Restoration requiring relatively low-latency access to large data sets and powerful computational engines and algorithms. Currently, these applications require relatively modest bandwidth from data storage, although requirements are likely to increase in time. Other applications, such as CM or DM are more latency tolerant but could require more substantial bandwidth. The development of algorithms and applications are being studied in other workpackages in this project.

4) Scalability

As networks grow in complexity and new measurement devices come on stream (capable of handling higher data rates), data storage and computational complexity will grow accordingly. Our architecture must be capable of offering scaling and upgrading of resources and capabilities to match the increasing computational and data needs.

5) Security

Security of the power system is its ability to keep functioning in the presence of malicious or accidental events which alter normal operations. Traditional power distribution systems are concerned mainly with safety and availability. Emerging smart networks will raise the security stakes further as all fundamental concerns will need to be addressed: safety, availability, integrity and confidentiality.

In the following, we will discuss security risks and possible mitigations for the HPC Engine and the Data Storage systems.

6) Model-based risk analysis

We use simple models (Figures 3a and 3b) to analyse the security risks. The models divide the overall distribution system into separate components. The DMS is a logical centre where all system components come together. The models illustrate how the number of vulnerabilities and security risks increase in a transition from traditional DNO systems to smart-grids.

- The traditional DNO system shown in Figure 3a consists of electricity network (cables, feeders, transformers etc.), DMS (distribution management system), MV communication network.[8]
- The smart-grid scenario is presented in figure 3b. The model is larger and it encompasses additionally smart-meters, HPC Engine and Data System. [9]

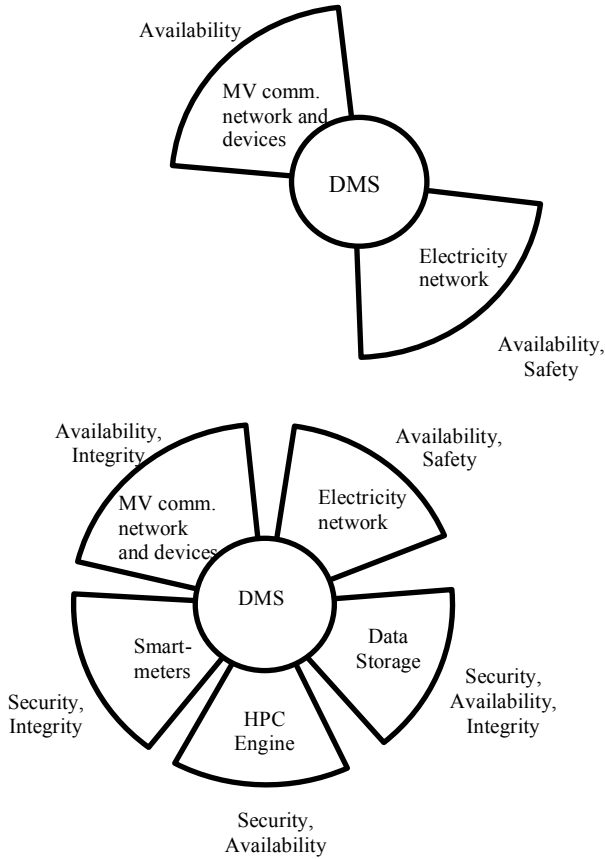


Figure 3 (a) Traditional Electricity Distribution security model, (b) Smart Grid security model.

The HPC Engine and the Storage Systems are physically located at the DNO, connected only to the Distribution Management System (DMS). The DMS is the only mechanism by which commands can be submitted into the HPC Engine and Storage Systems. There is mutual authentication between the DMS, HPC Engine and Data Systems. Security risks and mitigation strategies for the HPC Engine and the Storage System can be summarized as:

Security Concerns	Risks	Mitigations
Availability	HPCE failure HPCDS failure	Resilient hardware and software Fail-safe operations protocol Patching/upgrading policy
Integrity	Accidental data corruption Malicious data corruption	Data consistency analysis and validation
Security	Unauthorised access	Authentication and authorisation System isolation Access white-listing
All	Undetected security breach attempts	Logging Auditing Monitoring

The security of the communication infrastructures of the medium and low voltage networks, particularly with regard to any future AMI infrastructure, as well as the security of DMS is beyond the remit of the HPC system. However, some *post facto* mitigation strategies could be employed on the HPC side, to help minimise the risks of defective decisions being based on incorrect input data, such as data consistency analysis.

4. THE HPC DATA SYSTEM (HPCDS)

The Hadoop Distributed File System (HDFS) has been selected for the storage system [10]. HDFS is a highly fault-tolerant file system designed to run on commodity hardware. It is a part of a larger distributed computing framework, Hadoop [11], but can be used in other computational frameworks. HDFS consists of a logical hierarchy of files and directories physically spread across multiple networked machines, as shown in Figure 4.

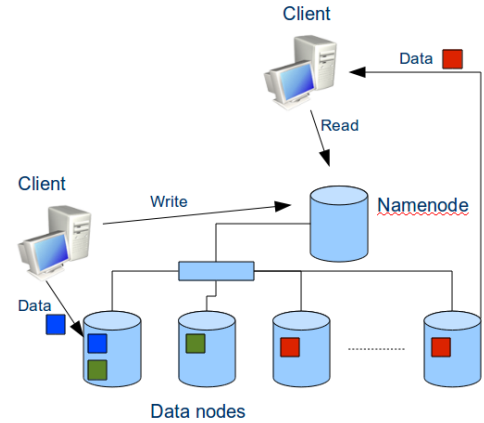


Figure 4: HDFS deployment example.

Storage and bandwidth capacity can be expanded by adding commodity servers to the network. The HDFS achieves scalability by decoupling meta-data operations guaranteeing high throughput [12]. The meta-data resides in the NameNode's memory for performance reasons.

Fault detection and automatic recovery are key design features of HDFS, using replication policies to specify the level of file-wise replication.

HDFS is optimized for high data throughput through multiple links rather than for low access latency. Files are physically split into blocks of defined size and distributed across the nodes. Low latency buffers at the data subsystem interface could be added to mitigate HDFS access latency.

HDFS is an open source software project, commonly deployed on Linux-based platforms and can use a range of different mechanisms, such as NFS.

Our design also required the complete decoupling of data storage from any data consumption and production agents. Data can only be accessed or stored via HPCDS client-server interface. Any agent requiring data access, such as an application running in the HPC or a DMS will invoke requests to the HPCDS-side server.

Other storage solutions could be used in place of HDFS, including SQL databases or hybrid systems, where file systems could be addressed via a database. The main point is that no ripple effects would be propagated to the rest of the overall DNO system.

5. THE HPC ENGINE

Figure 5 gives an overall view of the HPC Engine, of its components and of the relationship between them. Note the role of PELICAN in encapsulating computational tasks.

The DMS interacts with the HPC Engine only through a client (DMS) server (HPCE) mechanism. Requested executions are then coordinates by a commodity scheduler.[13] Many concurrent tasks, each of different priority, can be run at any given time, competing for HPC resources.

The interface can also receive DMS instructions such as revision of execution priorities, requests on current execution, etc. Only data and metadata specifying the computation can flow from the DMS to the HPCE; the HPCE must require all other data from the HPCDS (client-server). On task completion, the HPCE notifies the DMS. The DMS can then request from the HPCDS, manually or automatically, the results of the computation.

The interface must leverage available open standards such as the HPC Basic Profile ([14]) to describe the interface to the scheduler, in order to avoid vendor lock-in.

Existing tasks can be rescheduled, held or even cancelled according to the level of priority of newly submitted ones. This will ensure that the DMS will be assigning or reassigning tasks priorities rather than explicitly manipulating the underlying configuration of the queues.

The HPCE requirements are best met by providing a loosely coupled federation of commodity clusters. The largest cluster must be powerful enough to carry out the most intensive computations within the required "time window". Because of the requirements of security, efficiency, cost, fast interaction with the DMS and HPCDS, it is likely that the HPCE and HPCDS would be in close physical co-location.

The HPCE will be supported by a fast, local parallel file system, to access local and temporary data, wholly independent from the HPCDS. This was not explicitly included in the overall layout of figure 5.

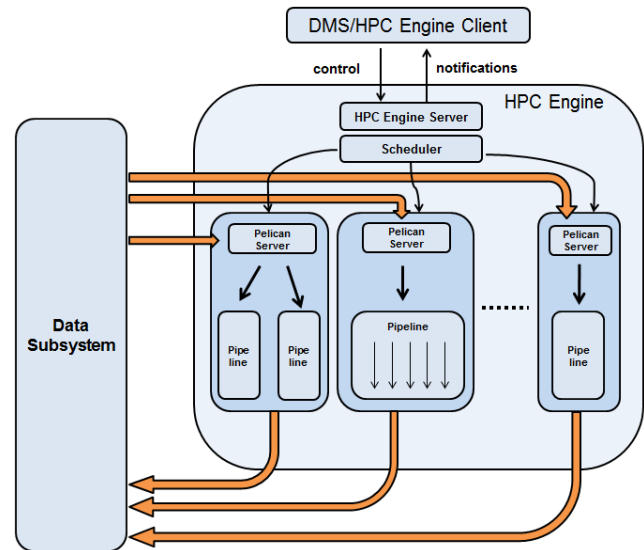


Figure 5: Overall view of the HPC Engine

A number of points were considered in this choice:

- **Availability, Cost, Proven Technology:** clusters are virtually ubiquitous, used in a wide range of environments with a wide choice of solutions, expertise and providers.
- **Resilience:** should a fault develop in a cluster, the scheduler would redistribute the load to the others.
- **Horizontal and Vertical Upgradeability:** the system could be upgraded when and if required with minimum impact on application and overall layout:
 - Horizontally: extra nodes could be added to the clusters or whole new clusters could be added to the federation.
 - Vertically: components of higher performance could replace existing ones or be added: for example, more recent CPUs, new technology such as GPUs, upgraded interconnects.
- **Flexible utilisation:** new applications or uses may require resources for development.. A federation of clusters could be split at any time into disjoint portions, and these could be recombined as necessity arises.

Section 2 defined the expected applications characteristics. Clearly they can all be mapped onto a federation of clusters, employing different types of parallel approaches: multi-threading (preferably OpenMP[15]), MPI[16] or hybrid (MPI + OpenMP). Highly parallel applications, such as DSE using the zonal approach, could also be handled through PELICAN, described below.

6. PELICAN

The PELICAN framework([17] is an efficient, light-weight C++ software library to process data streams in quasi-real time. The library aims to separate data acquisition from data processing,

allowing scalability and flexibility to fit a number of scenarios. With its origin in radio astronomy, it is currently used in a number of different applications (e.g. pulsar detection, fast imaging, etc).

Applications using PELICAN implement a number of components which adhere to a well defined API. Figure 6 (a&b) shows the overall view of PELICAN. We distinguish three main components:

- **Data server** -- a data server is a *process* not any specific item of hardware. Its function is to interface with the data stream and, if required split it in portions or chunks available to serve the pipelines.
- **Client** -- In the processing pipeline, this handles the data communication with the PELICAN data server. It is a component of the pipeline.
- **Processing pipelines** – These processes carry out the computation required. Pipelines do not communicate with each other. Any parallelism within the computation is entirely carried out within the pipeline. Pipelines may consist of several processes and/or threads executing in parallel.

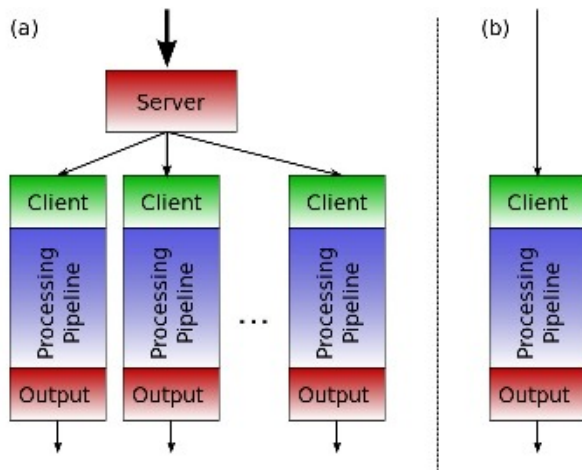


Figure 6 PELICAN framework overview

Figure 6b also shows that individual pipelines can connect through the client directly to an input data source, without requiring a data server (for example for discrete data processing).

This allows great flexibility in computation:

- Processing of continuous data streams. Servers and pipelines could run constantly, processing data as they are received. For example, it could be possible to keep the DSE running at all times. At scheduled intervals, data could be sent by the Data System to the PELICAN data server and this would initiate the computation.
- Modular approach and separation of input and processing. Each pipeline is highly modular. Also, the emphasis is independence from the input data source, whether it is a file, a stream or an instrument.
- Data encapsulation and separation of algorithmic content from the pipeline and PELICAN mechanisms.

Pipelines have highly modular structures. Modules allow as much as possible a “plug-and-play” structure for easy replacement of algorithms and also encapsulate algorithmic details thus hiding them the pipelines. Modules consist of a thin C++ interface to a

“computational engine” or algorithmic content. This could be delivered in any language that can be interfaced to C++ (e.g. C, Fortran, etc) to allow flexibility and reuse of existing software components. Processing pipelines themselves could be highly parallel, using MPI, OpenMP or both. Using the classification introduced above:

- **Serial, non-interacting applications.** Single serial pipelines with or without a PELICAN server, or multiple serial pipelines.
- Serial or modestly parallel non-interacting applications. This is no different then the case above.
- **Parallel applications with non-interacting processes.** A PELICAN server, including an appropriate “chunker” would be best. This also allows more freedom in the choice of algorithms etc: should a new algorithm incur in greater or lesser computational costs than existing ones, the number of pipelines could be varied without impact on the overall throughput. Should the data server recognise inadequate computational power, new pipelines could be started and these would automatically latch on the data server and share the load with the existing ones.
- **Fully parallel applications.** The sort of closely coupled parallelism can be provided within an individual, parallelised pipeline.

7. USE CASE: DISTRIBUTED STATE ESTIMATION

An important use-case consists of delivering at regular intervals the overall status of the distribution network. Upon request from the DMS the DSE is executed and the results are presented to the operator. (see Section 2.1).

Execution of the DSE on the HiPerDNO HPC platform consists of a number of steps:

- The request to compute the DSE is issued by the DMS (client) to the HPCE (server). Adequate meta-data describing specific computation may be included.
- DSE jobs have the highest priority, hence the HPCE scheduler provides the required resources, if necessary by stopping lower priority tasks and requeuing them
- The DSE (client) requests input data from the HPCDS (server).
- The results from the computation are handed over to the HPCDS.
- The HPC Engine notifies the DMS on completion of the DSE.
- The DMS retrieves the results from the HPCDS and presents them to the operator, either on manual or automatic request.
- The HPC Engine scheduler restarts the jobs interrupted earlier and/or starts new jobs in the queue according to their priorities.

8. SYSTEM SIZING

For a typical large distribution network, consisting of 80,000 nodes, we have analysed its requirements for:

- Processing power (size of cluster)
- Bandwidth (Data System only)
- Data Storage

Our estimates were based on data provided by HiPerDNO partners. As all codes are currently under development, we also had to make some assumptions, such as on the scalability of applications. It should be noted that our estimates are based on the performance of experimental codes and prototypes. We expect that fully optimized applications ported to the target platforms would perform considerably better.

Data refresh rates from measurement devices and the rates at which DSE, CM etc must be run exceed current practice and are in line with the DNOs expectations in the near to medium future.

AMI technologies are not directly examined by HiperDNO, and their HPC requirements are as yet unclear. These will clearly depend on the level of data aggregation required for LV networks, either technologically, commercially or legally.

The best algorithm for DSE is currently the “zonal” approach which is very highly parallel and very suitable for a framework such as PELICAN. It should achieve a good level of parallelism and load balancing. At the time of writing, stochastic algorithm would appear to require more resources. Collaborative work is currently underway between EDF France and the University of Oxford.

INDRA is studying a power restoration genetic algorithm. Their serial prototype could be parallelised across the members of a population (generation) and has high potential for achieving good scalability, given the relatively high cost associated with computing each member. Collaborative work between INDRA and the University of Oxford has also started.

Brunel University provided data about algorithms currently being developed for the CM of underground cables and transformers. This is currently been studied within the context of the HPC Platform.

At this stage of the HiperDNO project, the foundations and requirements of DM algorithms are being studied. They pose considerable challenges, as envisaged by the HiperDNO schedule. As the analysis of historical data is asynchronous with other applications, the requirements of DM algorithms could be incorporated at a later stage. It would also be a matter of DNO policies to allocate the resources for DM according to their specific time frame.

Overall bandwidth requirements for our large distribution network are modest, when taking into account the current and perspective measurement refreshing rates. Although no data for DM or PRA are yet available, in all likelihood they require data already needed by DSE and CM, respectively, so they would not add significant changes to the overall picture. The situation for AMI is more intricate: it is possible that legal and commercial requirements would only make aggregated data, available thus requiring modest bandwidth. Full data collection from all smart meters would increase bandwidth requirements very considerably indeed.

The following tables summarises the requirements:

Application	Number of Processors P	Bandwidth
DSE	> 22	< 20 Mb/s
PRA	> 10+	As DSE, in first instance
CM	> 3	< 20 Mb/s.
DM	See text	As CM, in first instance
AMI	See text	< 100 Mb/s (aggregated)

Storage poses a greater challenge. HiPerDNO partners reported that the data permanence requirements for the data could be between 2 to 3 years, but it is as yet to be finalised. This is summarized in the following table, in storage requirement *per annum*:

Application	Storage/Year	Comments
DSE	< 40 TB	
PRA	Minor extra costs	DSE data and other minor requirements, e.g. possible pre-computed restoration scenarios
CM	< 50 TB	1,000 devices, monitored every 0.1 seconds, 1,000 bits
DM	As CM	As CM with additional generated information (probably << CM)
AMI	< 350 TB	Assuming data aggregation

From the data available we could recommend that for the network considered the following:

HPC Engine	> 2 x 16-20-node (2-quad CPU per node) clusters
HPC Data System	> 1PB (> 2PB with full redundancy)

The two clusters would allow full redundancy as well as running background tasks, such as DM, advanced computation of restoration scenarios, testing software upgrades etc. Three clusters should be considered to provide a platform for extra computation as well as testing upgrades and patches: virtual machine technologies could also be used for this purpose, with the appropriate caveats.

9. CONCLUSIONS

We are specifying the following components for the HiPerDNO HPC and Data Storage System:

- Hadoop Distributed File System (HDFS)
- a federation of loosely coupled computational clusters
- the PELICAN computational framework

These fulfill the requirements that we have set out earlier in this paper and would provide an appropriate platform for the current challenges as well as those yet to come (such as AMI and the different modes in which this data can be used). In particular, we believe that this system would be capable of:

- Handling and storing heterogeneous data with varying refresh rates and bandwidth from different types of network measurement devices.

- Providing timely and simple access to data for:
 - DMS, when requested (including automatic requests),
 - The HPC Engine, when required by a specific computation
- Providing a computational solution for currently envisaged tasks:
 - High intensity, closely-coupled parallel computation (e.g. DSE, power restoration, global data mining), whatever their level and type of parallelism,
 - Large number of serial, non-communicating processes (e.g. CM, DM, etc).
- Scheduling computations of varying level of urgency:
 - Very high: power restoration,
 - High: e.g. DSE,
 - Medium: e.g. CM,
 - Low or Background tasks : e.g. DM (?) etc.
- Offering the required levels of flexibility and upgradeability as required by new and emerging distribution network technologies such as:
 - Smart metering (AMI),
 - New higher rate measuring devices,
 - New algorithms,
 - New methodologies, e.g. those required to deal with reactive power.

10. ACKNOWLEDGMENTS

The authors wish to thank the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number: 248135 for their support of the HiPerDNO research project[14].

The authors also thank all the members of the HiPerDNO consortium that have contributed to the research presented in this paper: Brunel University (UK); EDF SA (France); Fraunhofer IWES (Germany); GTD (Spain); IBM (Israel); Indra (Spain); Korona (Slovenia); EDF Energy Networks (UK); Union Fenosa (Spain); Elektro Gorenjska (Slovenia).

11. REFERENCES

- [1] Irving, MR., Robust algorithm for generalized state estimation, IEEE Transactions on Power Systems 24 (4) : 1886- 1887
- [2] K. Li, "State estimation for power distribution system and measurement impacts", IEEE Trans. Power Syst., vol. 11, pp. 911 - 916 , 1996.
- [3] Ali Abur, Antonio Gomez Exposito, "Power System State Estimation. Theory and Implementation.", CRC Press, New York (USA), 2004, ISBN 0-8247-5570-7.
- [4] Naka, S.; Genji, T.; Yura, T.; Fukuyama, Y.; , "A hybrid particle swarm optimization for distribution state estimation," Power Systems, IEEE Transactions on , vol.18, no.1, pp. 60- 68, Feb 2003
- [5] S. Nandi and H. A. Toliyat "Condition monitoring and fault diagnosis of electrical machine—A review", Conf. Rec. IEEE IAS Annu. Meeting, vol. 1, p.197 , 1999.
- [6] I. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 1999
- [7] Hart, D.G.; "Using AMI to Realize the Smart Grid," IEEE PES General Meeting, July 2008, pp.1 – 2
- [8] Electrical Power Research Institute (EPRI) 2010, "Distribution System Cyber Security architecture (Guidelines for Applying Security Measures to Meet Distribution Cyber Security Requirements)".
- [9] McDaniel, P., McLaughlin, S.: Security and Privacy Challenges in the Smart Grid. IEEE Security & Privacy Magazine (May/June 2009)
- [10] Konstantin Shvachko, Hairong Kuang, Sanja Radia, Robert Chansler "The Hadoop Distributed File System", Proceedings of MSST2010, 2010.
- [11] A. Bialecki, M. Cafarella, D. Cutting, O. O'Malley. "Hadoop: A Framework for Running Applications on Large Clusters Built of Commodity Hardware", <http://lucene.apache.org/hadoop/>, 2005
- [12] Konstantin Shvachko, "HDFS scalability: the limits to growth", ;login: 35, pp 6-16, 2010.
- [13] N. Capit, G.D. Costa, Y. Georgiou, G. Huard, C. Martin, G. Mouni, P. Neyron and O. Richard, A batch scheduler with high level components, IEEE/ACM Intl. Symp. on Cluster Computing and the Grid (CCGrid), IEEE Computer Society (2005), pp. 776–783.
- [14] Dillaway B, Humphrey M, Smith C, Theimer M, Wasson G. HPC Basic Profile, Version 1.0. Grid Forum Document GFD-R-P.114. 28 August 2007.
- [15] Leonardo Dagum , Ramesh Menon, OpenMP: An Industry-Standard API for Shared-Memory Programming, IEEE Computational Science & Engineering, v.5 n.1, p.46-55, January 1998
- [16] Message Passing Interface Forum, MPI: A message-passing interface standard. Internat. J. Supercomputer Appl. 8 3/4 (1994).
- [17] F. Dulwich, B.Mort, C.Williams, S.Salvini: "An Overview of PELICAN", in preparation (a preliminary copy is available from the authors of this paper).
- [18] F. Dulwich, B.Mort, C.Williams, S.Salvini: "PELICAN User Guide". A copy is available from the authors of this deliverable. Soon available from the OeRC website.
- [19] <http://www.hiperdno.eu>, HiPerDNO: High Performance Computing Technologies for Smart Distribution Network Operation, EU FP7/2007-2013 ICT Energy Funded Research Project