Secure, Performance-Oriented Data Management for nanoCMOS Electronics

[1]R.O.Sinnott, [1]C.Bayliss, [2]C.Davenhall, [3]B.Harbulot, [3]M. Jones, [1,4]C.Millar, [4]G.Roy, [4]S.Roy, [1]G.Stewart, [1]J.Watt, [4]A.Asenov
[1]*National e-Science Centre, University of Glasgow*
[2]*National e-Science Centre, University of Edinburgh*
[3]*North West e-Science Centre, University of Manchester*
[4]*Department of Electronics and Electrical Engineering, University of Glasgow*
United Kingdom
r.sinnott@nesc.gla.ac.uk

ABSTRACT

The EPSRC pilot project *Meeting the Design Challenges of nanoCMOS Electronics* (nanoCMOS) is focused upon delivering a production level e-Infrastructure to meet the challenges facing the semiconductor industry in dealing with the next generation of 'atomic-scale' transistor devices. This scale means that previous assumptions on the uniformity of transistor devices in electronics circuit and systems design are no longer valid, and the industry as a whole must deal with variability throughout the design process. Infrastructures to tackle this problem must provide seamless access to very large HPC resources for computationally expensive simulation of statistic ensembles of microscopically varying physical devices, and manage the many hundreds of thousands of files and meta-data associated with these simulations. A key challenge in undertaking this is in protecting the intellectual property associated with the data, simulations and design process as a whole. In this paper we present the nanoCMOS infrastructure and outline an evaluation undertaken on the Storage Resource Broker (SRB) and the Andrew File System (AFS) considering in particular the extent that they meet the performance and security requirements of the nanoCMOS domain. We also describe how metadata management is supported and linked to simulations and results in a scalable and secure manner.

Keywords: Andrew File System, Storage Resource Broker, nanoCMOS electronics, security, performance.

1. Introduction

The electronics industry is facing fundamental challenges in developing the next generation of electronic devices and systems. These challenges are caused by the decreasing scale of transistor devices which have now reached nano-scale dimensions [1]. This decreasing scale allows *in principle* to include more transistors on a chip thereby enabling larger, faster, cheaper circuits to be built – widely captured through Moore's law [2]. However, it is now accepted that previous assumptions on the uniformity of transistor devices no longer hold true [3-7]. The atomic dimensions of transistor devices introduce variability caused by the microscopic (atomistic) differences of the transistors including for example the number and positioning of dopants in the silicon. The scale of these devices means that fundamental (quantum level) physical properties of devices need to be considered to understand the differences between individual devices. In the presence of such intrinsic fluctuations, emphasis has shifted from predicting the characteristics of a single nano-scale transistor to predicting the statistical behaviour of ensembles of macroscopically identical but microscopically different devices. To address this demands revolutionary changes to be made in the way in which future integrated circuits and systems are designed to accommodate for the atomistic variability of devices.

The UK EPSRC funded *Meeting the Design Challenges of Nano-CMOS Electronics* (nanoCMOS) pilot project (www.nanocmos.ac.uk) is working in this space. In particular, nanoCMOS is looking at building a production level Grid-based infrastructure to address these challenges. We emphasize production level here from two key perspectives. Firstly the system must support extremely large computationally expensive statistical ensembles of device and circuit simulations [8,9] and subsequently manage the data and metadata associated with these simulations. These simulations are driven by a proliferation of data inputs and can generate hundreds of thousands of simulation result files and the meta-data associated with them. To deal with this, performance is one key consideration that has driven our data infrastructure developments. Secondly, novel device designs and their potential impact on systems design give rise to commercial valuable exploitation possibilities. Given this, security is paramount to nanoCMOS especially security of data and metadata associated with simulations [10,11].

In this paper we describe the data and metadata management requirements of the nanoCMOS project, and give an indication of the design flows between the tools used. We outline the performance and security oriented evaluation of SRB and AFS that was undertaken and discuss the results of this evaluation. We present the integrated system that seamlessly integrates data, metadata and security for the nanoCMOS researchers. Finally we draw conclusions on the work as a whole and outline areas of future work.

978-0-7695-3535-7/08 $25.00 © 2008 Crown Copyright
DOI 10.1109/eScience.2008.21

2. Data Management Requirements for nanoCMOS

To address the challenges facing the semiconductor electronics industry, the project has adopted a hierarchical simulation methodology. At the heart of this methodology is to support simulation of ensembles of statistically varying atomistic devices. This is achieved through an in-house electronics simulation code (Geronimo) as depicted in Fig 2.1.

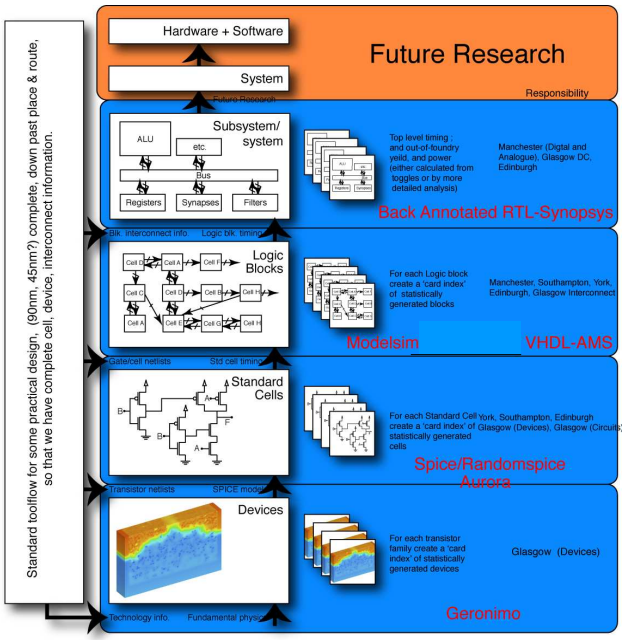

Fig 2.1: Tool based simulation methodology

This Fortran 90-based Geronimo application involves the solution of sets of coupled equations describing the distribution and flow of electrons in a given device structure. These might be based upon Drift Diffusion approaches solving Poisson's equation, or based on other approaches such as Monte Carlo based *ab initio* simulations. The application itself is computationally expensive with individual simulations typically taking in the region of 5-20 CPU hours on large scale HPC facilities such as ScotGrid (www.scotgrid.ac.uk). To correctly characterize the atomic variation of devices requires statistical ensembles of devices to be simulated. The input to this process includes dopant implantation information; oxide growth; etching, deposition of metals (typically provided by a commercial manufacturer such as IBM or Synopsis and hence of an extremely sensitive nature with IP value) as well as the simulation mesh to be used for the particular simulation.

Previously, ensembles of 200 or so devices were simulated (using slightly different dopant profiles) to characterize the behaviour of particular devices, however through HPC resources such as ScotGrid this has been increased to over 100,000 device simulations. We note that this characterization is typically given as the current/voltage (I/V) variation of a typical device. As well as current/voltage variations, the Geronimo application can generate a variety of other output data. For example, it can output data that is used for visualization of a given device. This might be to visualize the dopant distribution or the line edge roughness of particular devices. Typical outputs of a Geronimo simulation are shown in Fig 2.2.

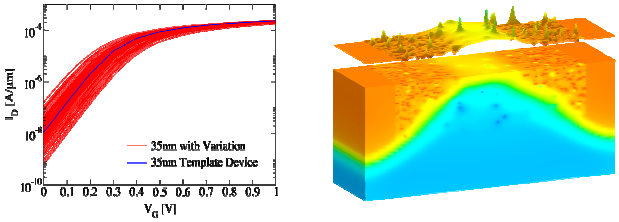

Fig 2.2: I/V variation (left) and Random Dopant Distribution (right) for 35nm transistor device

Having generated the set of I/V curves for a particular device, semi-empirical analytical descriptions of the response of a device are created: so called compact models. The process of generating a compact model is achieved through identifying the subset of device model parameters which most influence the curves. This process often involves exploitation of commercial tools such as Aurora™ which provides optimized curve fitting of device parameters to the generated I/V curves.

Once generated these compact models can be used by circuit simulators to predict the behaviour of circuits and systems built from multiple combinations of these compact models. Typical examples of the kinds of behaviour analysed at the circuit/system level with these compact models are to identify how the set of connected components respond to a stepped input voltage or to explore particular tolerances of the integrated circuit.

Feedback at this stage can require modifications to the generated compact models which in turn may require device simulations to be redone. Linkage of device simulations, compact models and circuit/system simulations are an essential component of the nanoCMOS project in understanding how atomistic variation of devices impact upon system level design and simulation.

Both commercial and non-commercial applications are used for circuit simulation. One of the most widely used circuit simulators today is the Simulation Program with Integrated Circuit Emphasis (SPICE) simulator which has both open source and licensed versions.

In all of these steps the management of data and meta-data is essential. Capturing the input parameters for device simulations; information on who ran the simulations and when; which I/V curves were used to generate which compact models and ultimately which circuits and systems were designed using these compact models needs to be seamlessly supported. The e-

Infrastructure in nanoCMOS must ensure the seamless linkage of this information for the distributed device modelers and circuit designers working in the project to understand, and ultimately address the impact of device variability in the design process.

There are many ways in which data can be managed on the Grid. The Open Grid Service Architecture Data Access and Integration Software (OGSA-DAI – www.ogsadai.org.uk) has developed generic middleware for access to and usage of data, typically in a service oriented architecture where the data is primarily stored in databases. The nanoCMOS project evaluated OGSA-DAI [9] however since the vast majority of data to be dealt with in nanoCMOS was file-based, an evaluation of two leading file based management systems in widespread use was undertaken: the Storage Resource Broker (SRB) [12] and the Andrew File Management [13] systems.

3. Overview of SRB and AFS

SRB was designed to provide a unified file system supporting access to files stored on a range of distributed file system back ends. The main SRB distribution is from the San Diego Supercomputing Center. SRB has been ported to a variety of UNIX platforms including Linux, Mac OS X, AIX, Solaris, SunOS, SGI Irix and Windows.

An SRB installation creates a global name space and allows the user to have a logical view of data across multiple physical, heterogeneous and distributed storage resources.

The SRB itself operates around a client-server model where data is stored on "vault" servers and metadata is stored on metadata catalogue (MCAT) servers. The client can request data to be copied between vaults as well as replicated. An SRB client can be a set of UNIX commands called *Scommands*, which have been developed to get the same comfort as in normal UNIX/Linux environment. Other access possibilities such as GUI interfaces also exist.

With regards to security, SRB supports federation between zones in a range of incarnations including single sign on and replication. Authentication is supported via several methods including the Globus Grid Security Infrastructure (GSI). SRB has been deployed across all of the core nodes of the UK e-Science National Grid Service (www.ngs.ac.uk).

Like SRB, AFS is a client-server based distributed networked file system. AFS uses a set of trusted servers to present a homogeneous, location-transparent file namespace to clients. AFS was originally developed as part of the Andrew Project at Carnegie-Mellon University, a distributed computing project which started 1983.

Through AFS, a user can log on and securely access a virtual file space crossing multiple heterogeneous resources. To support both structuring and security of this file space, AFS uses organizational units called cells where a cell can be considered as the collection of all the files belonging to an organisational unit. A cell may correspond to an actual organisation or, as is the case in nanoCMOS, a virtual organization. A cell comprises one or more servers and one or more clients. Each server, as might be expected, hosts a collection of files and makes them accessible throughout the cell. Each client allows access to the files hosted by the various servers. Underpinning the security of cells and hence AFS is the Kerberos security infrastructure [14].

AFS clients have also been deployed across the UK e-Science National Grid Service and resources such as ScotGrid. Given that both technologies are based on a client-server model and in principle provide secure access to distributed files, the nanoCMOS project in combination with the UK e-Science Engineering Task Force undertook a performance and security evaluation of these solutions.

4. Comparison of SRB and AFS

Within the nanoCMOS project two key requirements for comparison of SRB and AFS technologies were the overall performance of the technologies for access to and usage of distributed, file based data, and the security capabilities that they support to protect access to both file based data and the directories in which this data was stored. In terms of performance evaluation we focused in particular on access to and usage of the simulation results of the Geronimo application. These computationally expensive simulations can have a variety of results produced depending on the command line options selected: from a few *Kb* for I/V curve result data sets from Geronimo up to several *Gb* for data sets used for visualization. The majority of nanoCMOS data tends to be smaller result files however hence the overall performance in accessing and using results from potentially hundreds of thousands of smaller file-based data sets representing ensembles of microscopically varying devices is essential.

4.1. Performance Evaluation of AFS and SRB

When accessing and transferring files with the SRB or AFS there are many variables which might be considered for comparison, e.g. the number of local threads, remote threads, where checksums are calculated, the type of authentication used, the size of the files and the number of files transferred. We were especially interested in the time experienced by an end user to securely access a range of sizes of files across distributed HPC resources.

We note that in our performance evaluation we did not factor in performance delays caused by network latency since this was similar to both technologies. For the SRB evaluation itself we used version 3.4.0 of the SRB software and this was installed using the supplied script. The configuration of the Vault and MCAT was left as installed. Two different servers were used to host the SRB vault. Initially a quad 500MHz machine with 4GB of

RAM and a 250GB, 16 drive, SCSI RAID-5 array. This was later replaced with a dual Xeon 2.60GHz with 1.5GB of RAM and a 40GB IDE disk. The MCAT was hosted on the same machine as the vault using the version of PostgreSQL recommended by the installer script.

The first test measured the bandwidth efficiency or the ratio of bits in the source files and sent at the application level, of the SRB. This measurement is purely between the client and server omitting any network activity between the vault and the MCAT. This test was undertaken to explore the fundamental limits of the SRB. A high protocol overhead reduces the maximum available bandwidth of a link decreasing the apparent bandwidth.

Bandwidth usage was measured by recording the size of the payload of individual packets sent between the client and host using *tcpdump* during a transfer. This gave a measure of the total data transferred, omitting the TCP overhead. By comparing the total amount of data transferred with the on-disk size of the transfer the efficiency can be measured.

The SRB was found to make efficient use of bandwidth in single-threaded mode with the total TCP payload transferred within a fraction of a percent of the size of the data transferred. We note that this measurement excludes TCP/IP overheads such as enclosing headers and control packets. While the transfer is efficient the time taken is not. Traffic analysis shows a delay after the file transfer but before the connection is closed proportional to the size of the file being transferred which can increase the transfer duration significantly. The most likely cause of this is server side checksum processing being performed after the file has been completely downloaded. Server side CPU usage appears to spike during this delay and very little data is transferred after it, typically an ACK packet, one byte, a FIN and an ACK. After testing moved to the dual CPU system with more powerful processors transfer times improved.

The SRB client provides many tuning options for use in transfers. It may use the default serial mode, request up to 16 parallel transfers from the server or ask the server to select a number of parallel transfers. Similarly there is also batch mode intended for transferring multiple small files. Finally, there is full control over whether check sum calculations are performed on the client or server.

To measure the overall performance of SRB the metric used was total execution time of the *Sput* command as recorded by the *time* command. The transferred file was deleted using *Srm -f* between runs. Testing was automated using a small python script which ran each test option / file size combination 13 times. More repetitions were not used due to time constraints and the long transfer times of large files. For the same reason not every file size and option combination was tested. The test data was divided into small, medium and large file sizes with individual and collections of files for each class of file. The six classes of file were transferred using a selection of options. The number of client initiated parallel transfers was varied from 1 to 16, 16 being the SRB's maximum allowable, as well as serial mode. Other options experimented with included checksum calculation, batch mode and server initiated parallel transfers. In all the following results the recorded bandwidth has been expressed as a percentage of the benchmarked maximum between the client and server recorded using *iperf*. *iperf* was used rather than the quoted network bandwidth to provide an achievable comparison.

In undertaking the performance tests, three different file sizes were used: small (14.4MB); medium (52MB) and large (3.2Gb). The performance evaluation of SRB for small files is shown in Figure 4.1.

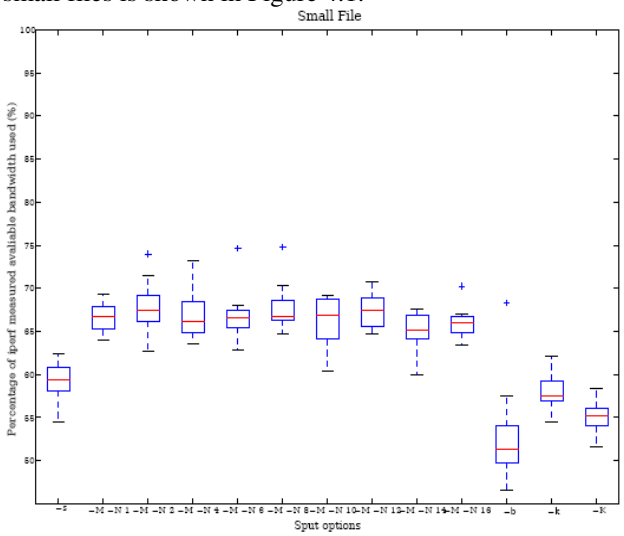

Figure 4.1: SRB Performance Results for Individual Small Files

As seen, the SRB performs poorly across all the test scenarios for small files with the best results coming from the client mediated parallel options , *-N <Number of threads>*, with the best performing scenario producing a mean of approximately 67% of the benchmarked score. The default serial transfer mode, *-s*, performs particularly poorly. Increasing the number of transfers does not significantly improve performance beyond the single threaded option. This is not an unexpected result assuming the SRB uses one thread per transfer as opposed to multiplexing the transfer across multiple TCP streams. The final two options shown, *-k* and *-K,* control whether the files checksum is calculated locally or remotely. In this test case the client side checksum *–k* performs better than the server side *-K*

The next test case used a group of small files each sized similarly to the small file from the previous test case totaling 300.8MB with their results shown in Figure 4.2. In this test case the relative performance between the different options is generally in line with the pattern from the small file shown previously. There is, however, a general improvement across the board over the previous

example of approximately 15% which is unexplained. As with the previous example moving from serial to a single parallel transfer results in an unexplained performance boost.

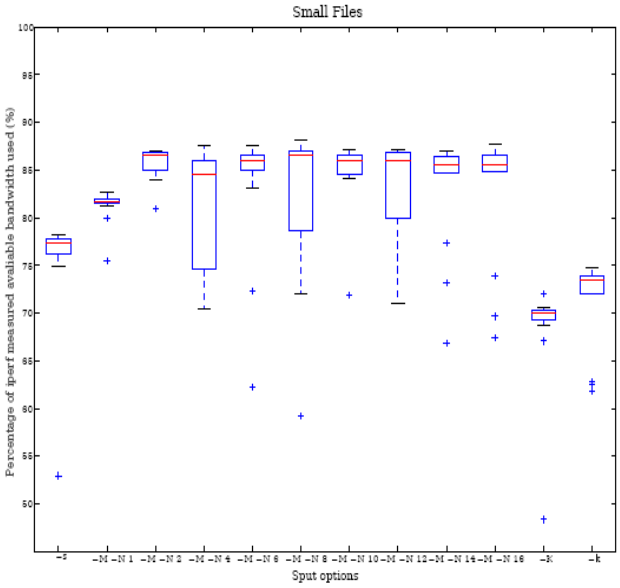

Figure 4.2: SRB Performance Results for Small File Groups

As expected with a collection of files moving to a truly parallel transfer results in a further performance increase over the single threaded transfers. The lack of increase beyond two threads is potentially due to the server only having two processors.

One interesting property is the increase in variability when the number of threads goes beyond 2 especially when the number of threads is divisible by 4. The mean performance remains in line but the performance can drop significantly. Quite why this would be the case is unknown it may merely be network contention as the client used a shared network to connect to the test server.

Finally, as with the previous example, using client side checksum calculation showed a significant improvement over server side checksum calculation.

For medium and larger sized files, the overall performance of SRB improved as shown in Figure 4.3 which shows the mean performance for all performance scenarios common to all test cases plotted against each other for comparison. The obvious features are the low performance of the small file test case, the high performance of the large file test case, the near universal poor performance of serial mode and the general pattern of increasing performance followed by a plateau. Figure 4.3 also shows that most test cases and scenarios produce results around the 80%-90% mark with four of the six test cases hovering in this band and only one producing consistently poorer performance.

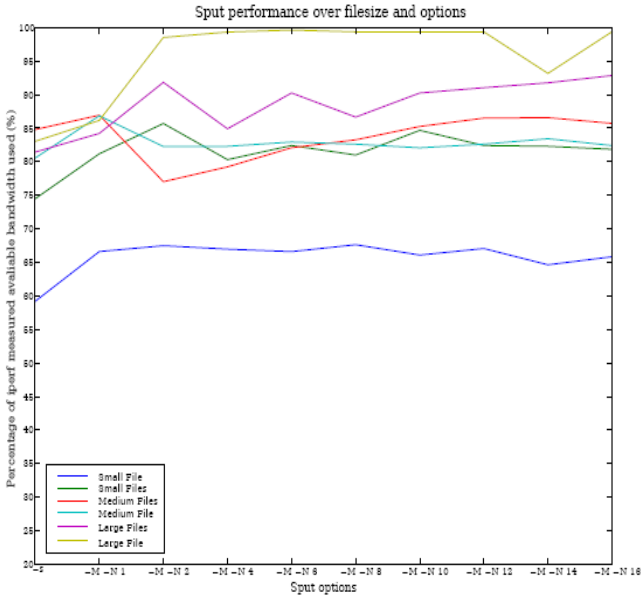

Figure 4.3: Comparison of Mean SRB Performance for Varying File Sizes

Based on the results of Figure 4.3 various conclusions can be drawn. Firstly, data transfer performance appears to vary with the number of threads employed. Unsurprisingly serial transfers offer the worst performance followed by single threaded transfers. Multiple threaded transfers generally exhibit further performance improvements however they also exhibit increased performance variation. This may be due to TCP congestion control being less problematic on low bandwidth links. Increasing the number of threads beyond 2 can be slightly detrimental to performance but still superior to serial threaded performance. The file size also appears to be a key performance impacting factor as the worst performing test case was the smallest and the best performing the largest. Serial transmission appears to operate differently from single threaded parallel as the latter provided universally superior performance.

The comparison in Figure 4.3 suggests that performance can be improved by favouring transferring single large files over multiple small files will increase performance. Likewise using one of the parallel transfer modes such as *-M 2* will likely improve performance but not in all cases. Finally, using client side checksum calculation should improve performance.

Given these considerations and the fact nanoCMOS has a direct requirement on fast access and use of hundreds of thousands of small file sizes, a performance evaluation of AFS was undertaken. We note that in this evaluation, the same file sizes were used as for SRB. We also note that in all of the performance test cases the client-server interactions in reading and writing data to AFS cells used encryption (using the AFS cell at NeSC established for the nanoCMOS project).

The first test conducted was to evaluate the overall throughput performance of AFS for access to a variety of files of varying sizes (using the same file sizes as for SRB). The results of this evaluation are shown in Figure 4.4.

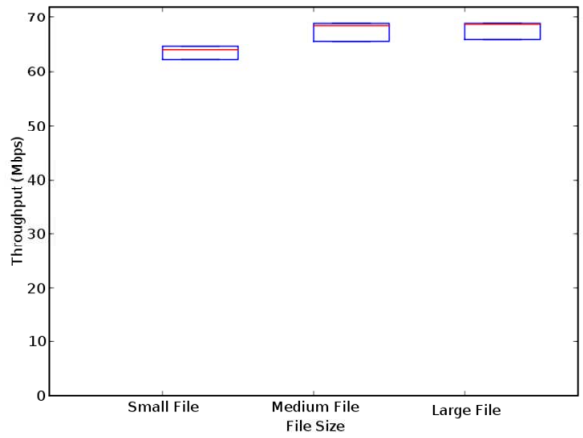

Figure 4.4: AFS Performance Results for Individual Files

Unlike SRB, AFS performance is not adversely affected by the size of files being accessed. Similarly, the overall performance with encrypted communications turned on is also good. Figure 4.5 shows the overall performance of AFS over various operations on a variety of file sizes. It also shows that reading and re-reading files performs better than writing/rewriting files.

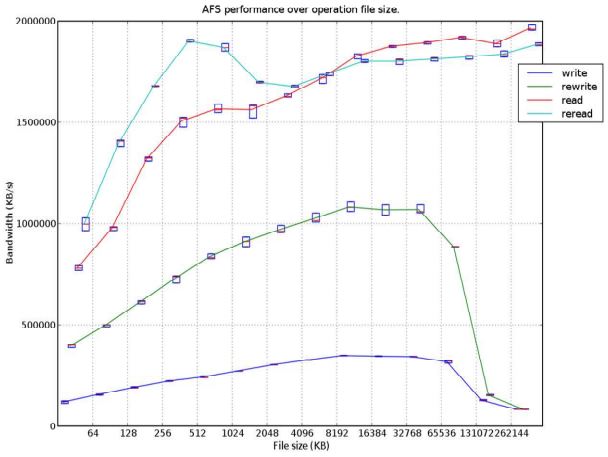

Figure 4.5: AFS Performance Results for Operation File Sizes

We note that reading and re-reading operations are the primary scenario to be supported in nanoCMOS since the simulations themselves generate many hundreds of thousands of files which are likely to be read *many* times and input to other tools in the electronics design tool chain, i.e. the files will typically be written once and read many times. One of the key requirements we need to support is to check before a simulation is run whether a computationally expensive simulation needs to be run, i.e. if the same devices with the same input files have already been simulated then repeated simulations are not required – unless of course these results were IP protected and not to be made available to given users.

Whilst reading/writing performance of distributed files is important for nanoCMOS, security is an essential factor that must be supported for adoption of the e-Infrastructure.

4.2. Security Comparison of SRB and AFS

SRB offers two forms of authentication; username and password combination and security based on X509 certificates. The X509 certificate option is available in several flavours including *GSI-Authentication, GSI-Delegation* and *GSI-Secure-Communication*. The username/password option is similar to login of a user to any UNIX/Linux like computer.

The basic *GSI-Authentication* option authenticates a certificate/proxy passed by the client and maps on to a username from MCAT, in a similar manner to the Globus grid-mapfile. *GSI-Delegation* allows the certificate/proxy to be used across servers whilst *GSI-Secure-Communication* provides support for data transport encryption.

Ultimately the level of expression and enforcement of security policies needed for authorization in SRB, i.e. not just authentication is limited. Rather, secure access to data stored in SRB is based around mappings from user X509 Distinguished Names (DN) to privileges set by an SRB administrator. Given the distributed nature of the data resources used in nanoCMOS, neither a single centralized administrator nor a grid mapfile-based approach, will meet the requirements of the nanoCMOS researchers or their industrial partners. Instead finer grained security is required which is more flexible and user driven.

Through use of AFS clients, nanoCMOS researchers are able to securely access and share data sets using Kerberos tokens. One of the key benefits of AFS with regards to security is that each AFS directory has an associated access control list (ACL). ACLs are typically administered by a directory's owner and/or local system administrators. An ACL comprises a list of entries prescribing who can access the directory and its contents and with what permissions. Each entry comprises a user or group and the permissions granted to that user or group. Groups are a collection of one or more named users. Users are able to create new groups, remove groups and to add individual users to groups, as well as the permissions they have, e.g. to create subdirectories, read and/or edit files etc.

Kerberos uses symmetric keys thus making it less computationally expensive than public key systems. It also has the advantage of offering greater security per key bit than public key systems. Unlike public key systems trust in Kerberos is established between entities at specific

hosts. This helps prevent stolen credentials being used on another host as the session keys are keyed to the host they were generated on.

Kerberos includes a federation system allowing transitive trust relationships to be established between realms. Furthermore, in PKI-based systems certificates become invalid when they expire or when they are added to the certificate revocation list (CRL). Given that a CRL needs to be updated regularly it is possible for compromised certificates to be used after they are technically revoked. Kerberos, through the enforced use of the KDC, is less susceptible to this since any session keys will usually expire within hours and only allow communication with a specific host.

The typical scenario supported in nanoCMOS for integrating X509 certificates (required for resources such as ScotGrid and the National Grid Service) and Kerberos is as follows. The user creates an X509 proxy credential either directly on their client machine or through a MyProxy service. Using this proxy credential, the user is able to select input files and simulations that they wish to run on compute clusters such as the AFS-enabled ScotGrid resource. When these jobs are submitted, the associated gatekeeper invokes the gssklog application to obtain the appropriate AFS tokens and the job is submitted and data staged to the appropriate AFS directories upon job completion. We note that gssklog is able to take an X509 based proxy certificate and authenticate with a gssklogd server. This server is then able to return appropriate AFS tokens. We also note that to support this scenario on Globus-enabled resources requires that gssklog is called before the globus-gatekeeper invokes any jobmanager –otherwise it will not be possible to write to the appropriate AFS directories. Of course, this scenario also mandates that appropriate AFS cells are established and importantly for cluster based usage, that AFS clients are installed on worker nodes.

One of the benefits of the AFS model of job submission over other approaches is that since we are working with a global file system, the notion of file staging to and from the cluster is moot. That is, the virtual directories can be considered as local to the AFS enabled cluster even when potentially remote.

5. Meta-data Management in nanoCMOS

To support the writing of data to secure directories or reading of data from directories, or indeed to support the electronics research process more generally, it is essential that the required metadata is associated with the results to facilitate their discovery, access and usage. The electronics domain has a variety of technology driven standards that are used to define the structure of data, i.e. these solutions have tended to be driven by the implementation of tools by tools vendors as opposed by international standards committees. This applies both to the result data formats as well as metadata formats.

To support integrated metadata and data management in nanoCMOS across the whole electronics design space, the project has implemented a novel metadata management model. In this model, results data and associated meta-data are both considered as resources, where a resource is an abstraction of any system entity. The main resources of the nanoCMOS data-management system are the input and output data of simulators such as Geronimo, as well as metadata associated with these files. Typical metadata that is captured when a given simulation is run includes the version of the simulator that was used; the input files used with the simulator; the resource on which the simulation sub-jobs were run and the person who ran the simulation itself. This information is captured by the metadata service through for example XML generated by the Geronimo simulator itself. We have deliberately designed the metadata service to be as flexible as possible, since the various researchers and tools that they use in the project is still being refined.

Independently of the mechanism used for accessing or representing a given resource, each resource is itself identified with its own Uniform Resource Identifier (URI). The use of URIs as identifiers also makes it possible to use the resources in a wider context, e.g. URIs to metadata can be passed around. Furthermore, URIs for metadata resources can include direct links to the source location for the actual data itself. Thus if a given simulation generates results which are stored in an AFS directory then access to that directory is dependent upon the client/user having the particular privileges, i.e. appropriate Kerberos tokens and being in the appropriate AFS group for example.

The metadata service has been implemented in Java, using the Restlet framework (www.restlet.org). The entry-point for a given client is an Apache Httpd server, used as a front-end to the Restlet container, to which Apache connects as shown in Figure 5.

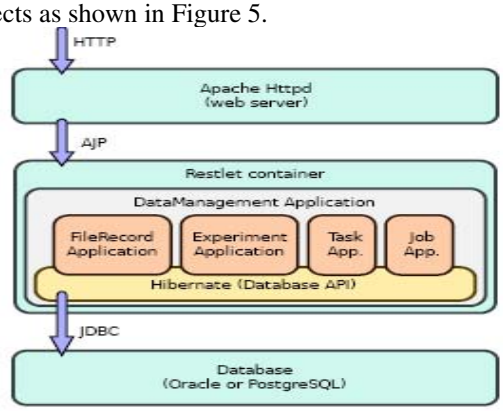

Figure 5: Metadata Service Components

As well as capturing and storing the metadata associated with simulations, the metadata service itself supports a

variety of query interfaces. These include options for querying particular technology nodes, e.g. IBM 45nm or Toshiba 35nm devices; for querying simulations run between particular time periods or by particular individuals. We note that whilst it is possible to run metadata queries, access to the actual data itself is dependent upon the user having the appropriate privileges. We note that refinements to the system such as restricting access to the metadata itself are currently being explored. We are considering two possibilities here: providing access control to the metadata service itself to restrict metadata access, or to establish one or more metadata services – one for publicly available data and the others for IP-protected data.

6. Conclusion

In this paper, we have described the requirements of the nanoCMOS project focusing in particular on the security and performance requirements on access to and use of file based electronics data. Through an evaluation with the UK e-Science Engineering Task Force we have shown that SRB has performance issues when dealing with smaller file sizes and when client or server side encryption is turned on. AFS offers both finer grained security possibilities and offers improved performance across a variety of file sizes with no degradation due to encryption. We have described how we have integrated the AFS into our Grid infrastructure and shown how metadata capture and linkage with secure file based data is supported.

The nanoCMOS project is still on-going and the work described here is still evolving. Many of the perceived requirements of the nanoCMOS researchers have changed since the project began, e.g. support for workflows is no longer regarded as essential in contrast to supporting very large scale bulk simulations and data management in a security-oriented setting. The work on nanoCMOS is also shaping many of the UK-wide Grid efforts including for example the deployment of AFS across the UK National Grid Service.

6.1 Acknowledgements

This work is supported by a grant from the UK Engineering and Physical Sciences Research Council (EPSRC). We are grateful for their support and for the inputs from the partners in the project as a whole.

7. References

[1] G.E. Moore, Cramming more components onto integrated circuits, Electronics, Vol. 38, No. 8, April 19, 1965.

[2] International Technology Roadmap for Semiconductors (ITRS), http://www.itrs.net

[3] D. J. Frank and Y. Taur, *Design considerations for CMOS near the limits of scaling*, Solid-State Electronics, vol. 46, pp 315-320 (2002).

[4] K. Takeuchi, R. Koh and T. Mogami, *A study of the threshold voltage variation for ultra-small bulk and SOI CMOS*, IEEE Trans. Electron Dev, vol. 48, p. 1995 (2001).

[5] A. R. Brown, A. Asenov, J. R. Watling, *Intrinsic Fluctuations in Sub-10 nm Double-Gate MOSFETs Introduced by Discreteness of Charge and Matter*, IEEE Transaction on Nanotechnology, Vol. 1 pp. 195-200 (2002).International Technology Roadmap for Semiconductors (ITRS), http://www.itrs.net/

[6] Khumakear R. at al., *An enhanced 90nm High Performance technology with Strong Performance Improvement from Stress and Mobility Increase through Simple Process Changes*, 2004 Symposium on VLSI Technology, Digest of Technical Papers, pp 162-163, 2004

[7] Wakabayashi H., *Sub 10-nm Planar-Bulk-CMOS Device using Lateral Junction Control*, *IEDM Tech. Digest*, pp. 989-991, 2003.

[8] R.O. Sinnott, A. Asenov, A. Brown, C. Millar, G. Roy, S. Roy, G. Stewart, *Grid Infrastructures for the Electronics Domain: Requirements and Early Prototypes from an EPSRC Pilot Project*, UK e-Science All Hands Meeting, Nottingham, UK, September 2007.

[9] L. Han, R.O. Sinnott, G. Stewart, D. Berry, *Towards a Grid-enabled Simulation Framework for nanoCMOS Electronics*, 3rd IEEE International Conference on e-Science, Bangalore India, December 2007.

[10] R.O. Sinnott, T. Doherty, D. Martin, C. Millar, G. Stewart, J. Watt, *Supporting Security-oriented Collaborative nanoCMOS Electronics e-Research*, International Conference on Computational Science, Krakow, Poland, June 2008.

[11] R.O. Sinnott, A. Asenov, C. Bayliss, C. Davenhall, T. Doherty, B. Harbulot, M. Jones, D. Martin, C. Millar, G. Roy, S. Roy, G. Stewart, J. Watt, *Integrating Security Solutions to Support nanoCMOS Electronics Research*, submitted to IEEE International Symposium on Parallel and Distributed Processing Systems with Applications, Sydney Australia, December 2008.

[12] Storage Resource Broker (SRB), http://www.sdsc.edu/srb/index.php

[13] R. Edward, R. Zayas. Andrew File System (AFSv3) Programmer's Reference: Architectural Overview, 1991.

[14] J.T. Kohl, B.C. Neuman, T.Y. T'so, *The Evolution of the Kerberos Authentication System*, Distributed Open Systems, pp78–94, IEEE Computer Society Press, 1994.