PaNdata-ODI

Deliverable D4.4

Benchmark of performance of the metadata catalogue

Grant Agreement Number	RI-283556
Project Title	PaNdata Open Data Infrastructure
Title of Deliverable	Benchmark of performance of the metadata catalogue
Deliverable Number	D4.4
Lead Beneficiary	STFC
Deliverable Dissemination Level	Public
Deliverable Nature	Report
Contractual Delivery Date	01 Jan 2014 (Month 27)
Actual Delivery Date	13 February 2014

The PaN-data ODI project is partly funded by the European Commission under the 7th Framework Programme, Information Society Technologies, Research Infrastructures.

Abstract

This document presents results related to the performance of the ICAT data catalogue and discusses their significance. Multiple benchmarking tests have been implemented. These were run by multiple facilities targeting local and remote ICAT servers. Various issues and performance factors related to metadata ingestion and querying, have been identified. The analysis of the results suggests an overall satisfactory performance.

Keyword list

ICAT, data catalogue, benchmarking, metadata, performance

Document approval

Approved for submission to EC by all partners on 13 February 2014.

Revision history

Issue	Author(s)	Date	Description
0.1	Milan Prica, George Kourousias	18 Dec. 2013	First draft version
0.2	Milan Prica, George Kourousias	20 Jan. 2014	iGest ingestion and search results on Elettra local
0.3	Milan Prica, George Kourousias	6 Feb. 2014	iGest benchmarking from multiple parties in SV7
0.4	Milan Prica, George Kourousias, Alistair Mills	7 Feb. 2014	Corrections
1.0	Milan Prica, George Kourousias	12 Feb. 2014	Final version

Acknowledgements: Alistair Mills, Rolf Krahl

Table of Contents

1. Performance and Benchmarking of Data Cataloguing Services

This document is a study on the performance of data cataloguing as is implemented in ICAT. The performance metrics / benchmarks are of synthetic nature. Such benchmarks make assumptions about the higher-level mixed processes that inherit the characteristics of the systems in-use. The systems consist of hardware and software infrastructure. Additional factors like network performance affect the results. Regarding the software, the benchmarks are based on the code of iGest (designed in D.4.2 and deployed in D.4.3). This includes different data ingestion and search scenarios on instances of ICAT v.4.3. The study notes that there are practical limitations on measuring performance as not all scenarios can be covered. Moreover the ingestion and searching may have different implementations and deployments so the results may vary. Note that the examination is mostly for core ICAT systems focused on metadata and their retrieval. Optional services like the IDS which downloads actual data files have been tested but are beyond the scope of this document. Nevertheless the design of the tests has been such that the investigation provides useful indicators that promote the discussion of the broader issues that affect the performance of cataloguing.

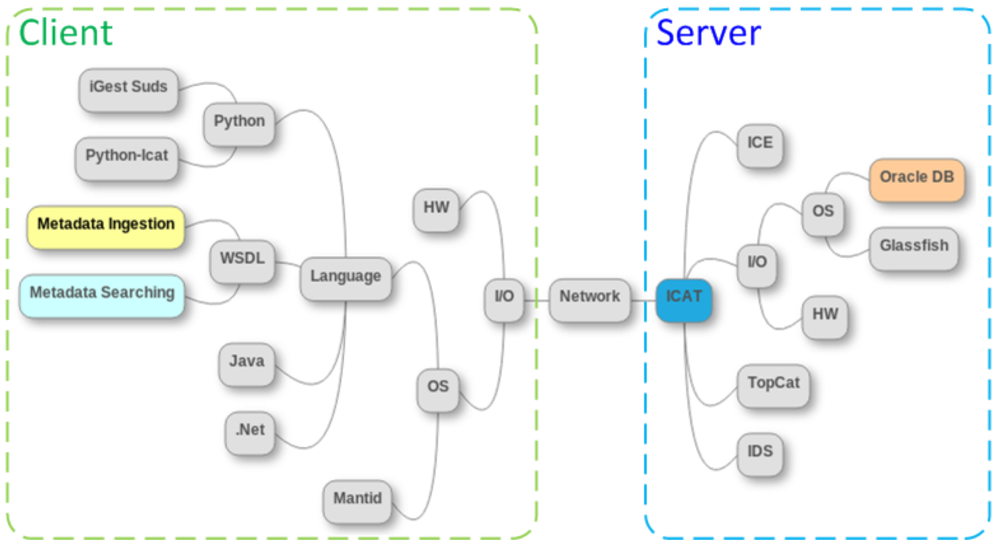


Figure 1. An ICAT server-client architecture involves multiple components with different performance issues

2. Hardware and Software

The underlying layers that a cataloguing system is based on have a significant effect on its performance. The hardware choice may affect both the Client that ingests (iGest) and the Server that catalogues (ICAT DB). In facilities like those of the PaNdata consortium, the ingestion is often part of the acquisition system that is based on a distributed control system (Figure 2). The searching is more often done by an end user terminal (i.e. laptop). As expected, there are plenty of different cases but the hardware in the CPU, the memory and the network infrastructure should be considered of high importance. Multiple setups both for clients and servers have been tested.

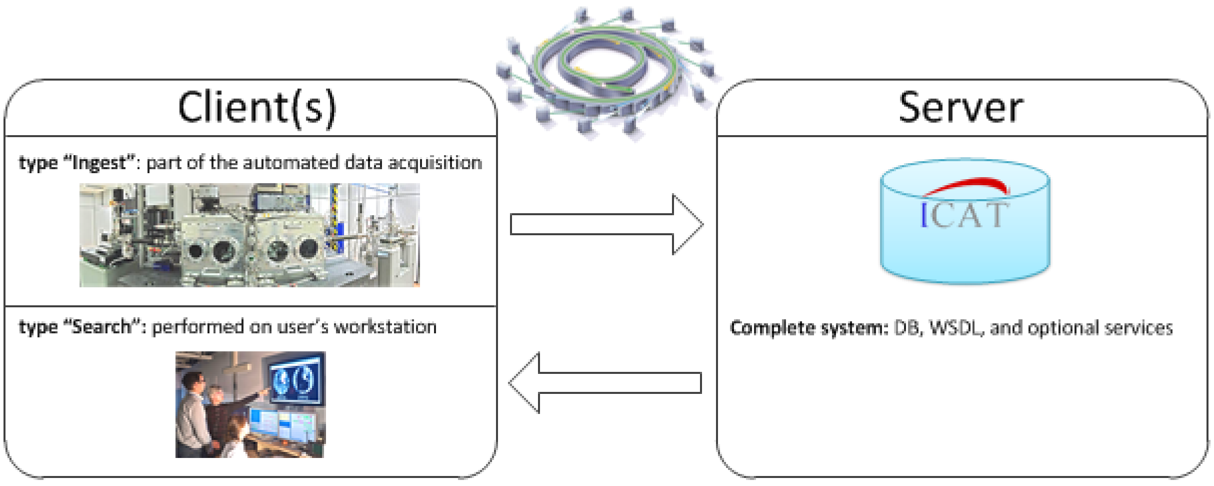

Client(s)
type "Ingest": part of the automated data acquisition
type "Search": performed on user's workstation
Server
ICAT
Complete system: DB, WSDL, and optional services

Figure 2. Facility ICAT with clients of different roles

The software in the D4.4 case is the latest ICAT to date (v.4.3.2) deployed in Linux and tested by iGest scripts on Linux clients as well. iGest is in Python 2.7 while ICAT's DB is Oracle. A critical component is the ICAT API as exposed in WSDL. The measurements are performed by custom scripts based on iGest, enhanced with various timing functions. Suitable Python Decorators[1] store the results for later interpretation to a file. The measurement overhead is minimal. Standard Python profiling (cProfile) has been performed but the information is of limited use due to the external Suds/WSDL calls.

3. iGest Benchmarking

The benchmarking tests are based on the code iGest[2]. iGest follows the design of an ingestor as described in D4.2 and implemented in D4.3. Other approaches to ingestion exist (i.e. Python-Icat[3]) and optimised implementations are possible if developed for custom cataloguing scenarios. Nevertheless, the modest size and its reference implementation, render it a suitable code base for prototyping and testing. While the original iGest generates sample Virtual Lab NeXus files, the benchmark scripts[4] exclude their generation so that the local I/O is part of the measurement. In addition there is also an adapted version of the iGest benchmark that uses the Python-Icat module

[1] https://wiki.python.org/moin/PythonDecorators
[2] http://pandata.org/igest/
[3] https://code.google.com/p/icatproject/wiki/PythonIcat
[4] https://code.google.com/p/pandata/source/browse/#svn/trunk/ops/sv7/client/test_igest

rather than direct Suds[5]. For completeness purposes, the time required to generate a Virtual Lab NeXus file with the required metadata but without actual data is approximately 15ms. An early observation was the overhead during ingestion caused by querying the catalogue whether the item to be ingested already exists (e.g. `Facility, Instrument`). This has been rather obvious in tests where complete ingestion cycles are performed (`Facility,...,Datafile`). For simplicity the test excludes setting of authorisation Rules.

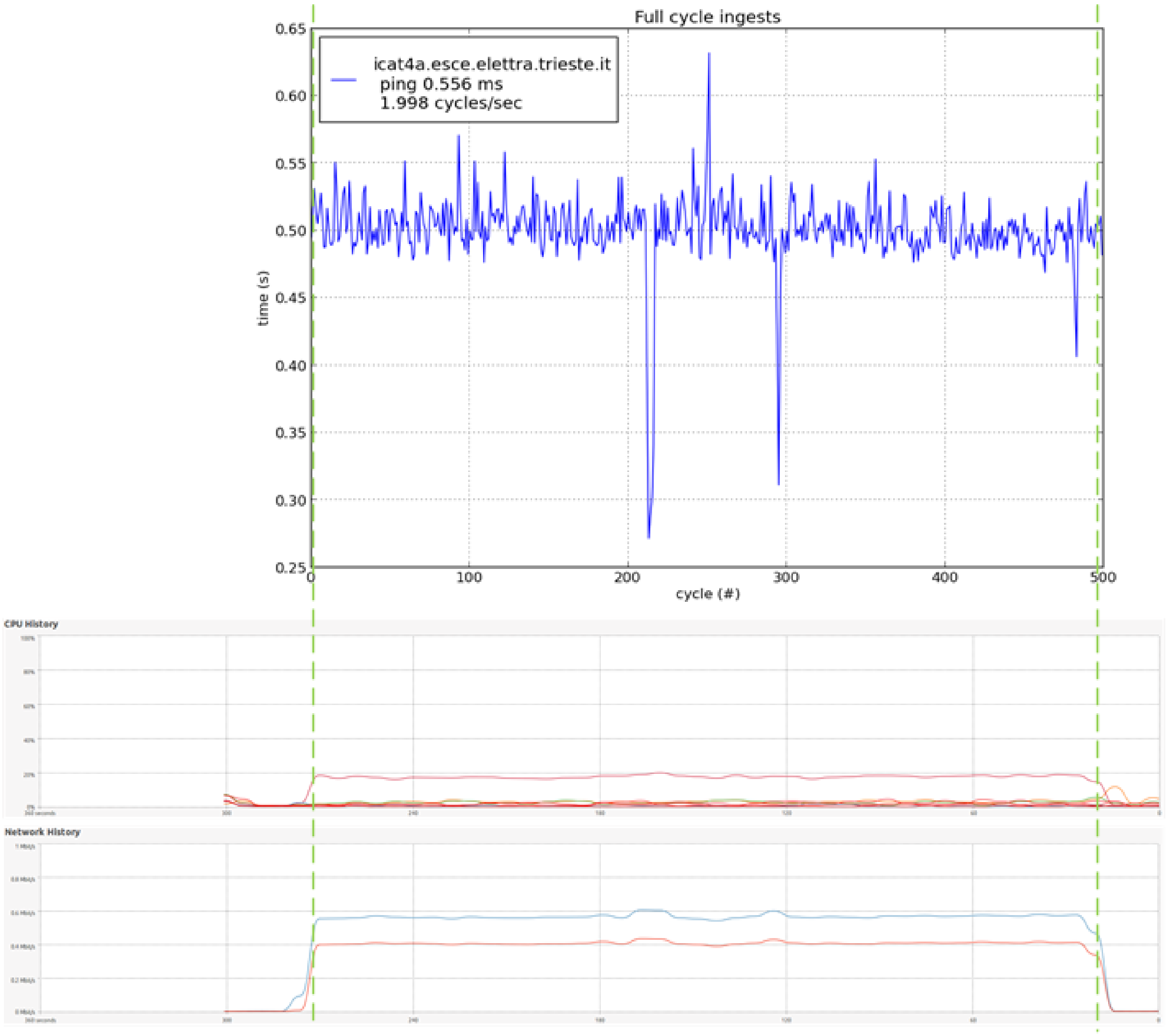


Figure 3. 500 ingestion cycles in local ICAT: timing, CPU and Network[6]

[5] https://code.google.com/p/pandata/source/browse/#svn/trunk/ops/sv7/client/test_igest-python-icat

[6]The labels on the axis are as follows. CPU History vertical axis labels: 0%, 20%, 40%, 60%, 80%, 100%. Network History vertical axis labels: 0 Mbps,0.2 Mbps, 0.4 Mbps, 0.6 Mbps, 0.8 Mbps, 1 Mbps. Horizontal axis is divided into 60s sections.

Figure 3 shows a single iGest client performing 500 full ingestion cycles (`Facility,...,Datafile`) into a local (same network) ICAT. The network graph (blue line: receiving, red line: sending) shows that even in the case of ingestion the client receives more data than it sends. The CPU graph demonstrates the utilisation of a single core of the test computer[7].

4. Concurrency in Cataloguing

The results of this study suggest that both ingestion and searching can substantially benefit from parallelism using concurrent execution. The previous deliverable (D4.3) has already hinted that cross-facility searching when performed in parallel returns the results more quickly. A simple configuration where the list of facilities actually contains the same facility yields a substantially higher number of queries per second. The benchmarking test is extended to examine ingestion as well and when executed currently it has a near linear increase in performance. Note that above a certain number of concurrent operations, the Glassfish server used by ICAT reaches its limit and further performance improvements are not possible, although optimization may overcome this issue.

[7] Intel i7-3820 @ 3.60GHz with 24GB RAM (Ubuntu 12.04)

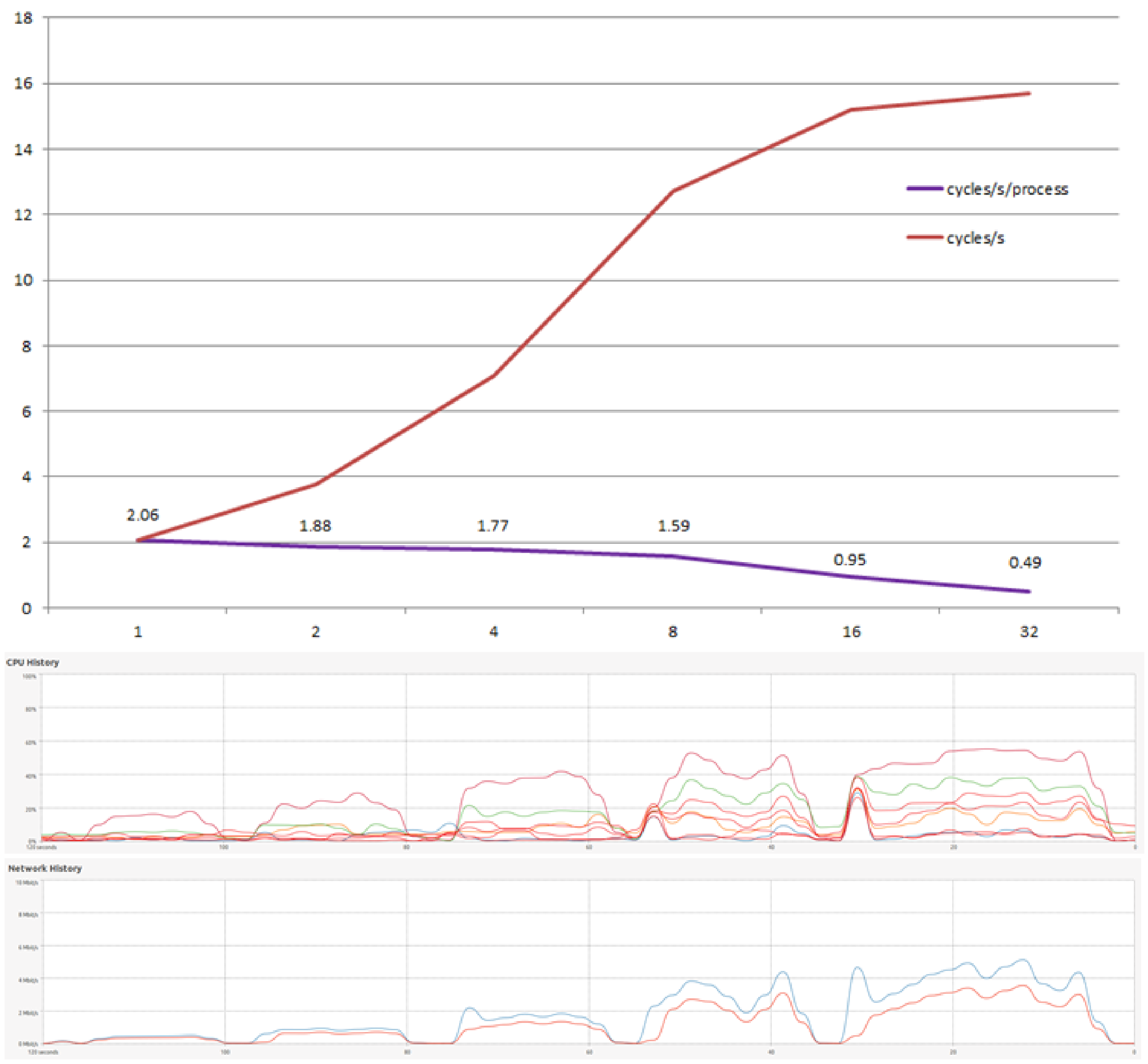


Figure 4. Performance for parallel full ingestion clients (1, 2, 4, 8, 16, 32). CPU and Network for 1,2,4,8,16.[8]

The first test was for the type of concurrency where multiple complete cycles of ingestion (`Facility,...,Datafile`) were performed. As shown in Figure 4 the parallel approach scales successfully with the number of processes (red line). The CPU and Network graphs (note they are only till 16 clients) show multicore use and the expected increase in dataflow. Nevertheless for the 16 concurrent ones, both CPU and Network show an extended period suggesting declining in performance. At the same time the server[9] saturated its CPUs at 32 clients. Each process performed 100 full ingestions (`Facility,...,Datafile`).

[8] The labels on the axis are as follows. CPU History vertical axis labels: 0%, 20%, 40%, 60%, 80%, 100%. Network History vertical axis labels: 0 Mbps, 2 Mbps, 4 Mbps, 6 Mbps, 8 Mbps, 10 Mbps. Horizontal axis is divided into 20s sections.

[9] 3 Core VM (Intel Xeon E5-2670@2.60GHz) CentOS 6.2

Concurrent ingestion may not only be used for enhanced performance but reflects the realistic scenario where multiple instruments ingest data at the same time in the facility catalogue (Figure 2). In a similar manner, multiple users can search the catalogue for data. A suitable test has been performed (Figure 5).

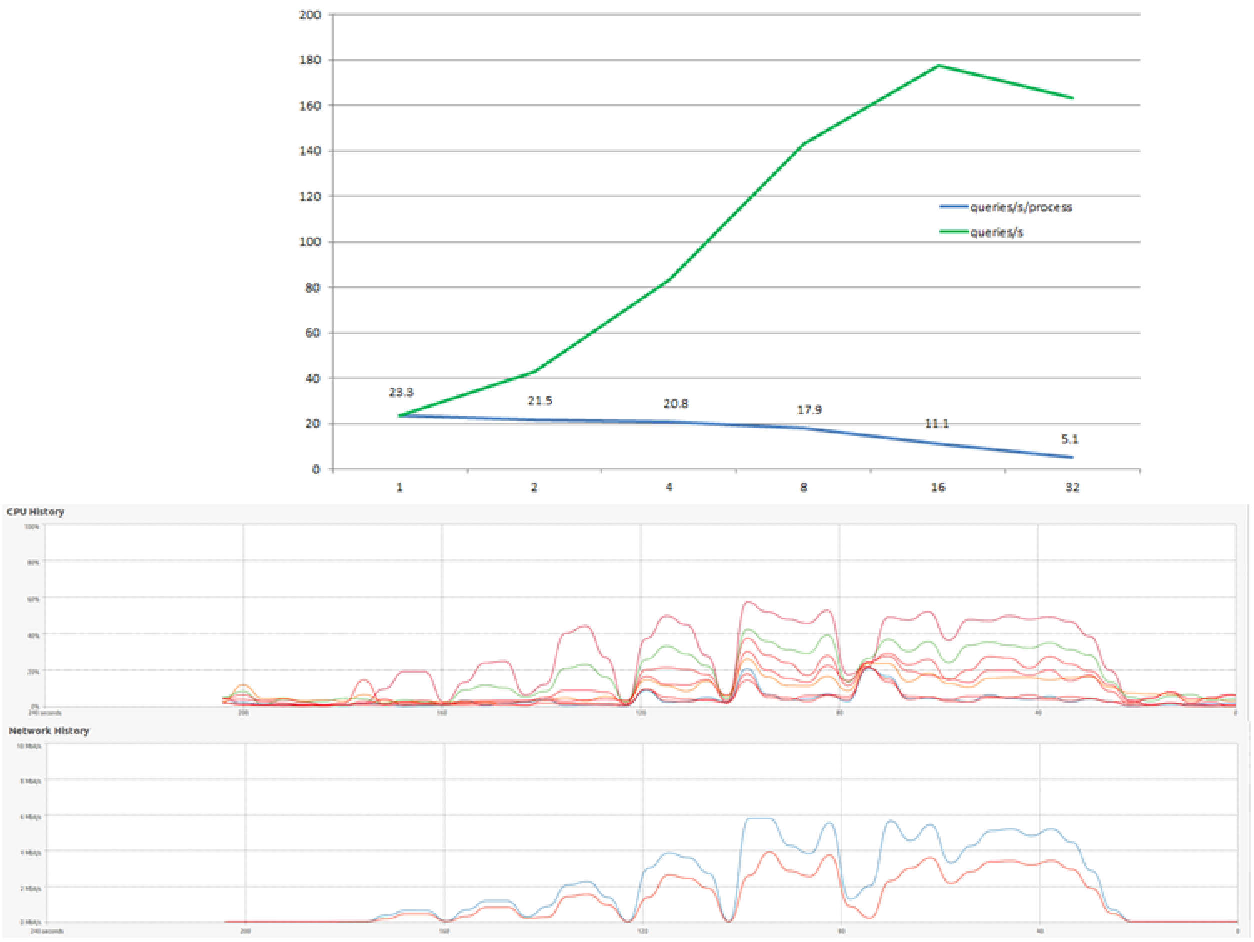


Figure 5. Performance for parallel search clients (1, 2, 4, 8, 16, 32) including CPU and Network[10].

The parallel searching clearly demonstrated that above 16 concurrent searches, the performance decreases. This is of course for the specific setup of the tested systems. Note that each single process performed 200 sequential searches.

5. Fast ingestion of multiple DataFiles

The complete cycle ingestion demonstrates the overall process but this section also examines the performance of a more realistic ingestion processes where an ICAT element (e.g. `Datafile`) is

[10] The labels on the axis are as follows. CPU History vertical axis labels: 0%, 20%, 40%, 60%, 80%, 100%. Network History vertical axis labels: 0 Mbps, 2 Mbps, 4 Mbps, 6 Mbps, 8 Mbps, 10 Mbps. Horizontal axis is divided into 40s sections.

ingested assuming that its required parent elements (e.g. `Facility,...,Dataset`) already exist. This naturally is much faster and represents the case where a fast acquisition system catalogues `Datafiles` on an already existing `Facility` to `Dataset` hierarchy (Figure 6). In this case, the `Datafile` ingestion is 8.5 times faster (Figure 3) compared to the complete cycle (even when the `Facility,…,Dataset` exist there is still a cost for querying their presence). On the Network graph of Figure 6 there is Inbound traffic (receive) peak that reflects a single query which returns the 2000 `Datafile` identifiers.

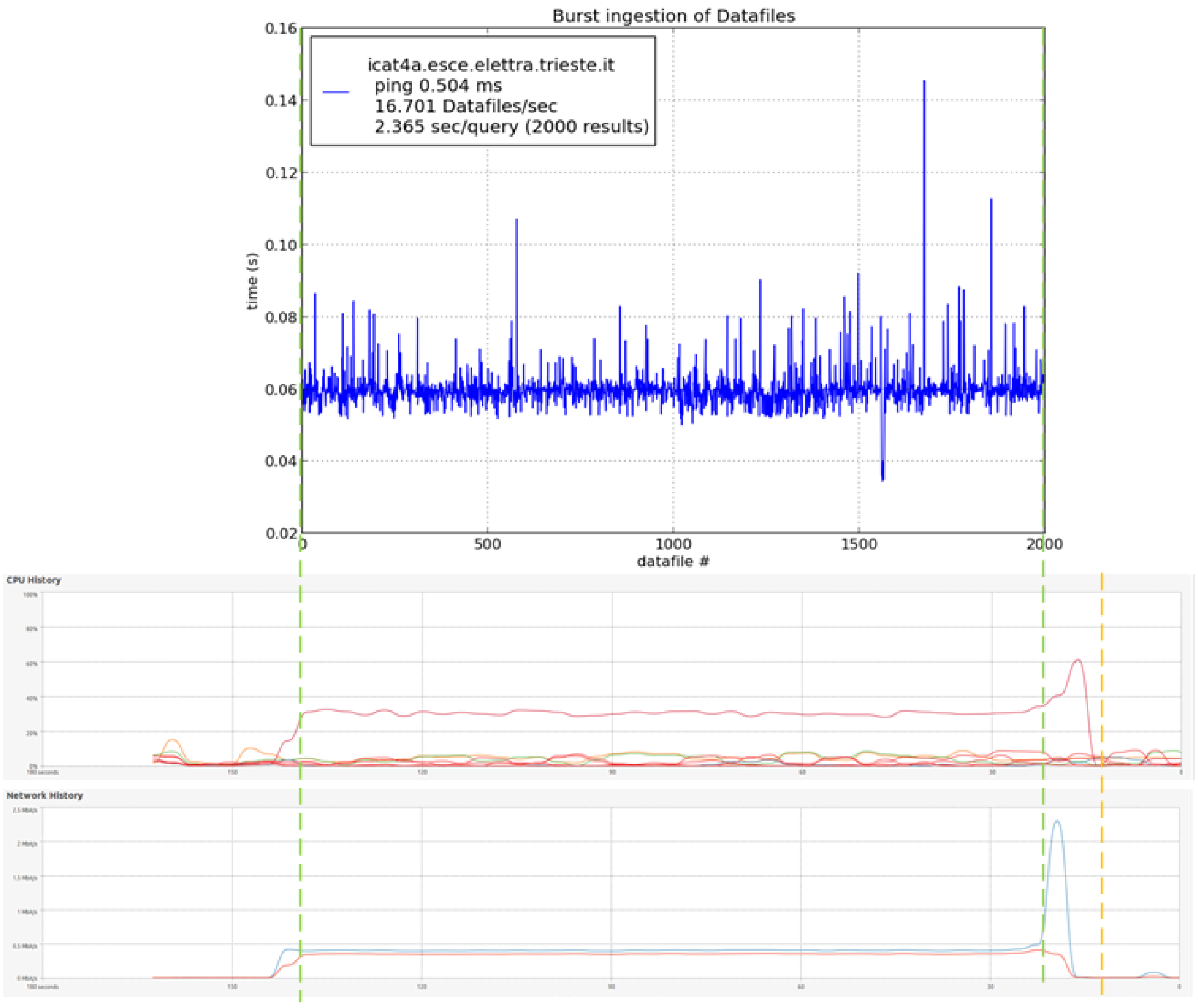


Figure 6. Burst ingestion of Datafiles assuming uniqueness and existing parent elements, followed by a single query that returns all of them[11].

[11] The labels on the axis are as follows. CPU History vertical axis labels: 0%, 20%, 40%, 60%, 80%, 100%. Network History vertical axis labels: 0 Mbps, 0.5 Mbps, 1 Mbps, 1.5 Mbps, 2 Mbps, 2.5 Mbps. Horizontal axis is divided into 30s sections.

6. Local and Remote Catalogues

The characteristics of the Network and its performance have an impact on ICAT. High bandwidth may be required for parallel processes while faster transmission to reception (round-trip time) with low packet-loss, affects high frequency burst ingestion operations. The Network charts of Figure 4 and Figure 5 reflect the volume increase of ingestions and searching. The following section on benchmarking from different facilities clearly shows that when the client is operating on a local ICAT endpoint (e.g. Elettra ingesting in icat-elettra.grid.elettra.trieste.it), it is much faster than when it is connected to a remote one (Figure 9).

7. Benchmarking in SV

A benchmark has been developed and included the scheduled Service Verification 7. It is based on iGest and performs metadata ingestion and catalogue searching. Since it is executed by multiple partners/facilities, it reflects the performance variations due to different hardware, software and network.
The tests were performed from the following nine PaNdata facilities:
ALBA, ELETTRA, DLS, ILL, ISIS, HZB, PSI, SOLEIL, STFC.

There were two flavours of the iGest benchmark: the main one was based on Python-Suds and the other on Python-Icat.

There were five different ICAT instances used:

- icat-elettra.grid.elettra.trieste.it
- tng-5.default.stfc.uk0.bigv.io
- icat.helmholtz-berlin.de
- www.pandata.org
- wwws.esrf.fr

Not every facility/client performed the tests against all targets and not everyone used both the Suds and the Python-Icat versions.

The benchmark includes four tests:

1. multiple complete ingestion cycles
2. multiple queries that return single results
3. single ingestion of multiple `Datafiles` in burst

4. single query that returns multiple results

This test also reported on additional information including host OS and network transmission to reception (round-trip time) from Client to ICAT endpoint. The pinging often failed due to firewall restrictions.

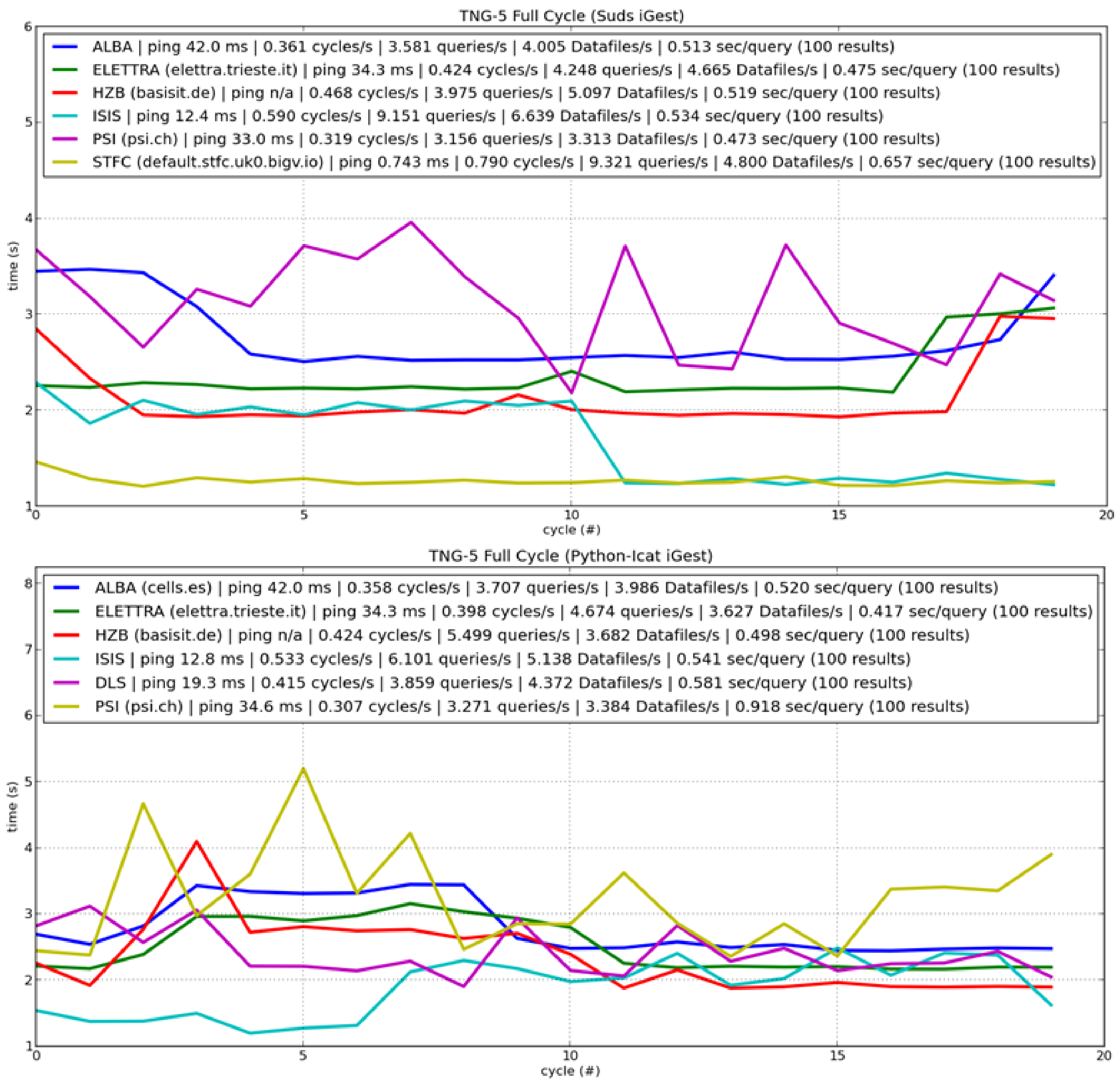


Figure 7. iGest benchmark (Suds/Python-Icat) from multiple facilities targeting the TNG-5 sever

The data illustrated in Figure 7 were collected on a test pointing at an ICAT deployed on the BigV cloud hosted in the United Kingdom. STFC ran the Suds client from a host on the same cloud while ISIS ran both clients from a different network in UK. The physical proximity may be an explanation of their performance. A similar case is in Figure 9 where both ESRF and ILL perform equivalently well on the ICAT of ESRF.

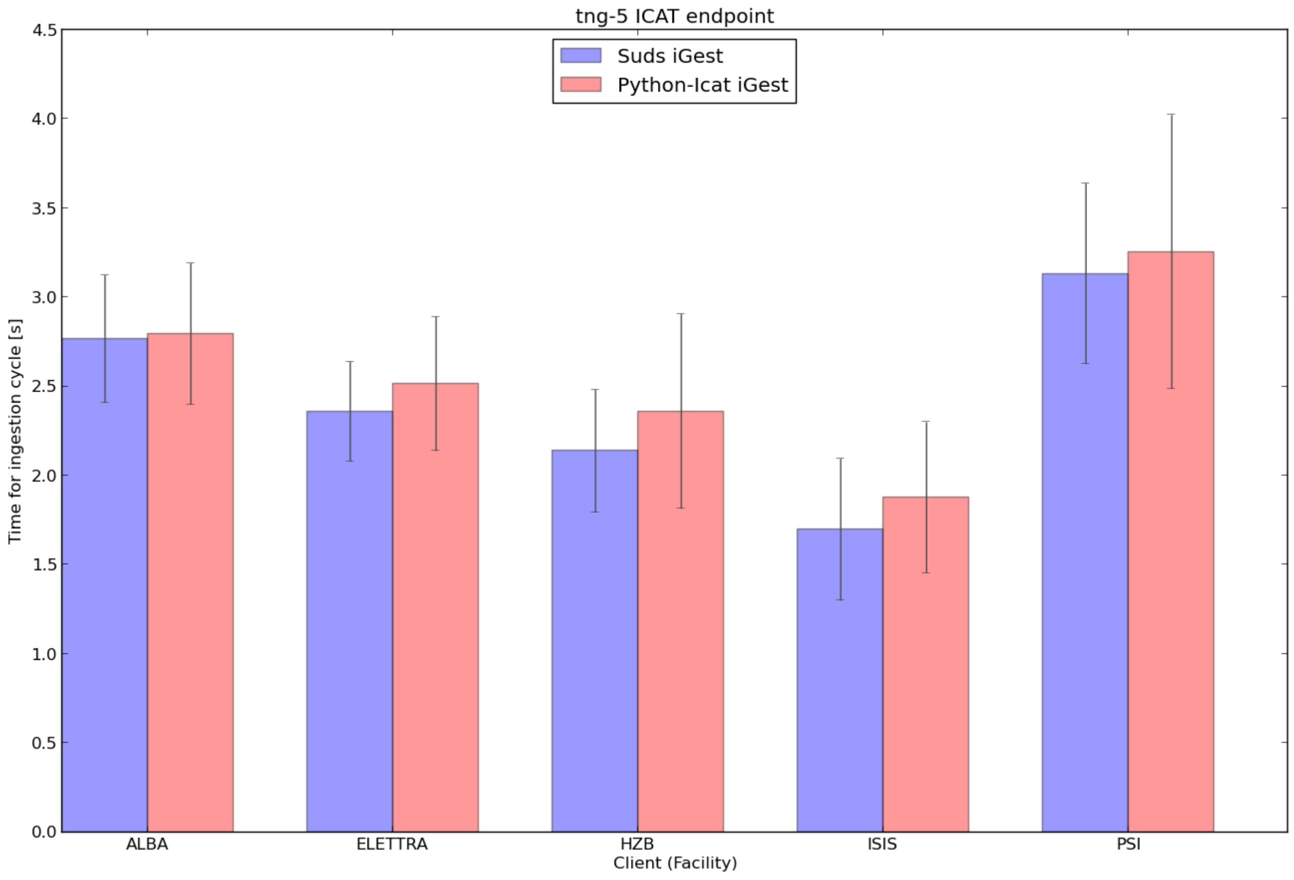


Figure 8. iGest Suds and Python-Icat test for mean value and standard deviation for 20 cycles in TNG-5

The data of Figure 7 and Figure 8 hint on the performance (and its stability) of different ingestion implementations. Each has its merit and both perform well.

On Figure 9 the charts show ICAT servers hosted by partner facilities. The local clients (i.e. *Elettra* → `icat-elettra.grid.elettra.trieste.it`) show a significant performance advantage over the remote ones.

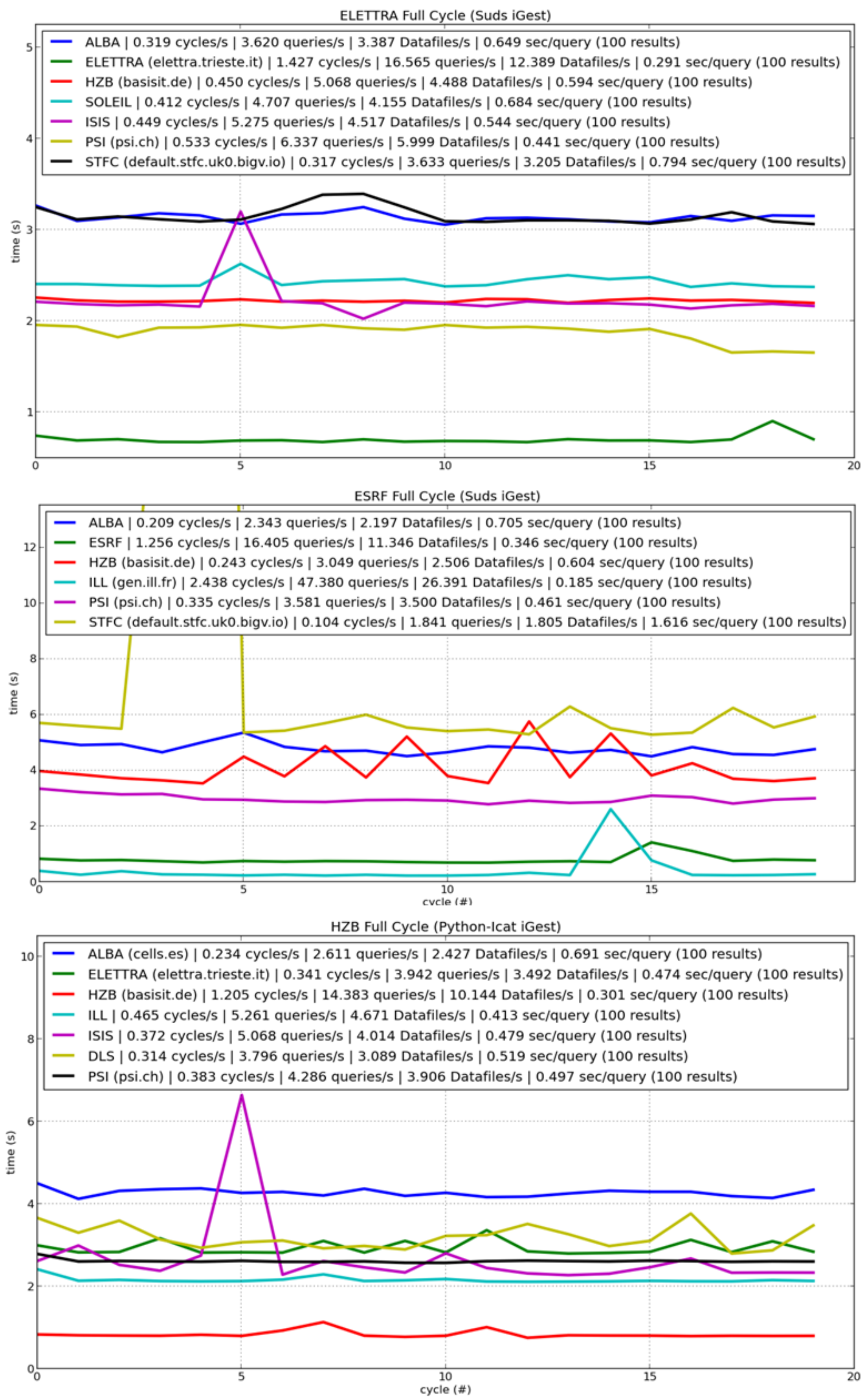


Figure 9. iGest benchmarks for ELETTRA, HZB, ESRF hosted ICATs

8. Conclusions: Discussion and Observations

For large and complex systems like data catalogues, measuring their performance is a substantial task. A data catalogue such as the examined ICAT relies on multiple software and hardware subsystems all of which may have different performance characteristics and requirements. As expected, for most of the networked based deployments the actual network itself is an important determinant of performance. Moreover, ICAT has multiple modalities of use, for instance ingestion of metadata and searching functions. Each of them may have different performance characteristic and factors that affect them. In D4.3, ICAT has been approached as an “ecosystem” to describe loosely that a data catalogue includes elements beyond the expected database and the core APIs. Such elements are user-front ends (e.g. TopCat, ICE), web portals, integration to existing applications (e.g. DAWN, MANTID), and additional services like data converters and data access systems (e.g. IDS). ICAT deployments may differ in terms of provided services as each PaNdata partner may have different requirements. The study excluded benchmarking add-on components like TopCat, ICE and IDS, and authentication mechanisms like Umbrella, as these are optional and beyond the scope of general evaluation of the core ICAT. Nevertheless for specific deployments it is recommended to examine those as well. The benchmark was based on iGest, a reference non-optimised implementation suitable for prototyping and studying ingestion and searching. The nature of the test is synthetic (e.g. complete ingestion cycles) rather than focused on isolated components (e.g. Oracle DB). Custom solutions may yield better results but also caveats. An example has been the attempt to ingest by bypassing the ICAT API/WDSL and to operate directly in DB (SQL) level. The performance has been much better; nevertheless this can easily result in incoherence and eventually an incorrect data catalogue. Due to the importance of the data that the PaNdata facilities produce, it is not recommended to perform ingestions this way in spite of the performance advantage. This study examined the latest ICAT 4.3.2 but older versions are still in production.

The principal conclusion is that ICAT is a well performing cataloguing system. Both ingestion and searching are fast enough for purpose. ICAT can scale well thus it can satisfy the needs of a range of facilities, from small to very large. As ICAT is based on a stack of software and hardware components, it can benefit from fine tuning (e.g. network and Oracle DB). It should be mentioned that high frequency instruments like the Free Electron Laser (FEL) FERMI @ELETTRA (data rates of 50Hz), require special approaches to data cataloguing as its requirements are not met by the above-mentioned benchmarking statistics. The positive verdict on the performance of ICAT is supported by this set of tests, the long term use of ICAT at the facilities, and a series of Service Verifications which were part of the PaNdata-ODI project.

Appendix: SV6 report

WP4 has been reporting on the progress and results of the Service Verification actions. D4.2 and 4.3 have included such SV reports (0-5) relevant to their Description of Work. The present Deliverable had tests performed as part of latest SV7[12]. All the relevant results (benchmarking) are presented in this document. For continuity/completeness purposes and for the benefit of PaNdata partners using the ICAT system, this Appendix includes the previous SV6 report. It does not contain information relevant to the task of D4.4.

Service verification 6 started on November 1st, 2013 and completed by mid-November.

Of particular note in this service verification were the ICAT services of ISIS and DLS; this is the first time these production quality services have been part of a service verification. A number of services appeared for the first time in this SV:

- Verification of provenance data in ICAT..
- Inclusion of a Fuseki service test.
- Deployment of Nexus support on client nodes. This performed well where Nexus was deployed correctly. Some partners had difficulty in deploying the software.

This service verification included a further, successful test of the European Affiliation Database.

The collection of ICATs is now quite diverse, with the ICATs being a mixture of ICAT 4.2 and ICAT 4.3, a mixture of Glassfish 3 and Glassfish 4, a mixture of databases such as Oracle, MySQL and Derby. The results of the tests were not affected by this diversity. The three factors which prevented 100% success were a firewall setting, self-signed certificates and the use of non-standard ports for services.

A comparison with previous SV shows that the project is making progress with higher participation, higher success rates, more services, and higher value of the services.

There were six parts to the test, and ten of the twelve partners provided a response: Alba – Spain; DESY – Germany; DLS – UK; Elettra – Italy; ESRF – France; HZB – Germany; ISIS – UK; PSI – Switzerland; Soleil – France; STFC – UK.

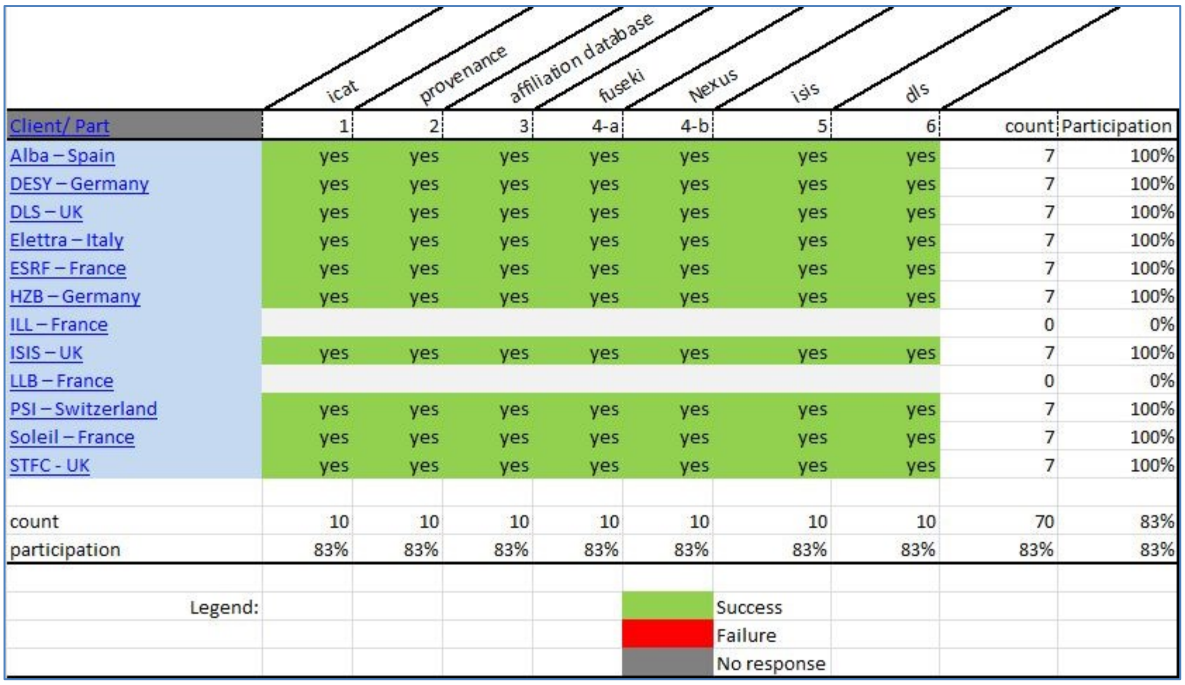
Eight of the twelve partners provided an ICAT service: DLS – UK; Elettra – Italy; ESRF – France; HZB – Germany; ISIS – UK; PSI – Switzerland; Soleil – France; STFC – UK.

STFC provided the Fuseki server, and the provenance records in an ICAT.

ESRF provide the European Affiliation Database.

The participation of the partners in the client tests is shown in the Figure A.

[12] The SV7 results regarding additional services, will appear on: http://pandatawp4.wordpress.com/service-verifications/report7/

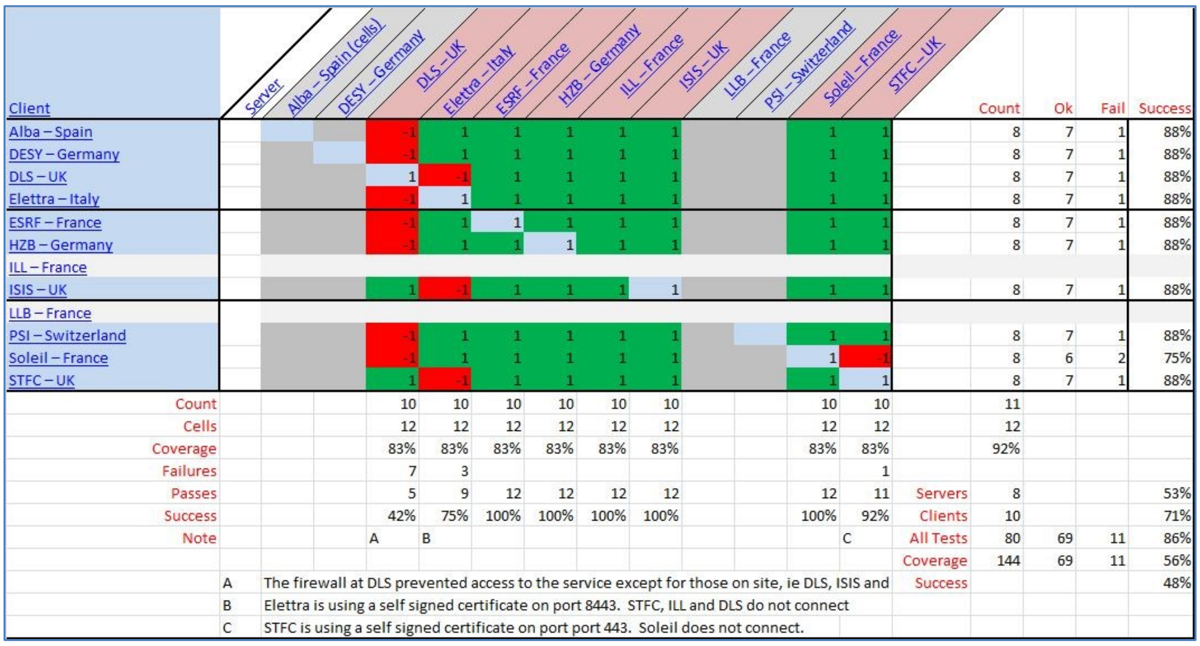


	icat	provenance	affiliation database	fuseki	Nexus	isis	dls		
Client/ Part	1	2	3	4-a	4-b	5	6	count	Participation
Alba – Spain	yes	yes	yes	yes	yes	yes	yes	7	100%
DESY – Germany	yes	yes	yes	yes	yes	yes	yes	7	100%
DLS – UK	yes	yes	yes	yes	yes	yes	yes	7	100%
Elettra – Italy	yes	yes	yes	yes	yes	yes	yes	7	100%
ESRF – France	yes	yes	yes	yes	yes	yes	yes	7	100%
HZB – Germany	yes	yes	yes	yes	yes	yes	yes	7	100%
ILL – France								0	0%
ISIS – UK	yes	yes	yes	yes	yes	yes	yes	7	100%
LLB – France								0	0%
PSI – Switzerland	yes	yes	yes	yes	yes	yes	yes	7	100%
Soleil – France	yes	yes	yes	yes	yes	yes	yes	7	100%
STFC - UK	yes	yes	yes	yes	yes	yes	yes	7	100%
count	10	10	10	10	10	10	10	70	83%
participation	83%	83%	83%	83%	83%	83%	83%	83%	83%
Legend:				🟩	Success				
				🟥	Failure				
				⬛	No response				

Figure A: SV6 Participation

SV Results

Parts 1 and 2 – Run a client script which collects information from the ICAT services. The results are summarized in the Figure B.



Client	Server	Alba – Spain (cells)	DESY – Germany	DLS – UK	Elettra – Italy	ESRF – France	HZB – Germany	ILL – France	ISIS – UK	LLB – France	PSI – Switzerland	Soleil – France	STFC – UK		Count	Ok	Fail	Success
Alba – Spain				-1	1	1	1	1	1			1	1		8	7	1	88%
DESY – Germany				-1	1	1	1	1	1			1	1		8	7	1	88%
DLS – UK				1	-1	1	1	1	1			1	1		8	7	1	88%
Elettra – Italy				-1	1	1	1	1	1			1	1		8	7	1	88%
ESRF – France				-1	1	1	1	1	1			1	1		8	7	1	88%
HZB – Germany				-1	1	1	1	1	1			1	1		8	7	1	88%
ILL – France																		
ISIS – UK				1	-1	1	1	1	1			1	1		8	7	1	88%
LLB – France																		
PSI – Switzerland				-1	1	1	1	1	1			1	1		8	7	1	88%
Soleil – France				-1	1	1	1	1	1			1	-1		8	6	2	75%
STFC – UK				1	-1	1	1	1	1			1	1		8	7	1	88%
Count				10	10	10	10	10	10			10	10		11			
Cells				12	12	12	12	12	12			12	12		12			
Coverage				83%	83%	83%	83%	83%	83%			83%	83%		92%			
Failures				7	3								1					
Passes				5	9	12	12	12	12			12	11	Servers	8			53%
Success				42%	75%	100%	100%	100%	100%			100%	92%	Clients	10			71%
Note				A	B								C	All Tests	80	69	11	86%
														Coverage	144	69	11	56%
	A	The firewall at DLS prevented access to the service except for those on site, ie DLS, ISIS and												Success				48%
	B	Elettra is using a self signed certificate on port 8443. STFC, ILL and DLS do not connect																
	C	STFC is using a self signed certificate on port port 443. Soleil does not connect.																

Figure B: SV7 Summary

Part 3 – Run a script which locates information in the European Affiliation Database.
The partners all reported success with this test. The team at ESRF did a very good job of preparing the test.

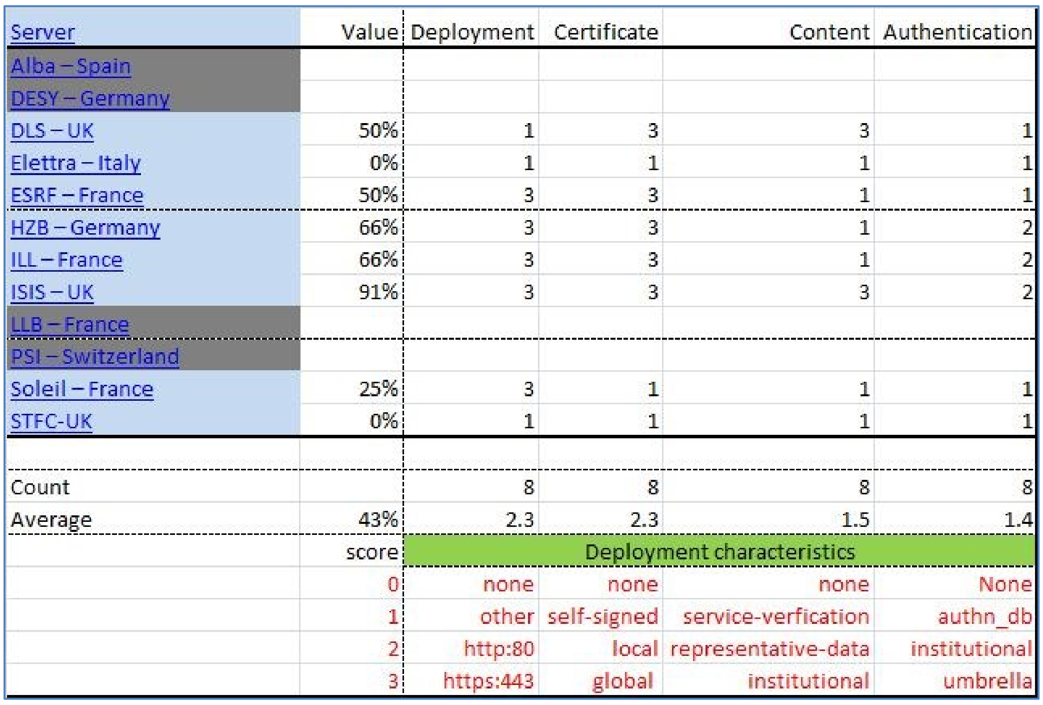
Part 4 – Run a script which retrieves information from the Fuseki server and from a Nexus file.
All of the partners were successful in retrieving information from the Fuseki server. Some of the partners were successful in retrieving information from a Nexus files, as they had difficulty installing the Nexus client software.

Part 5 – Run a client script which collects information from ISIS.
All of the partners were successful.

Part 6 – Run a client script which collects information from the DLS.
The partners within the firewall at Rutherford Appleton Laboratory were successful with this test. The firewall prevented access for the others.

Development of the Pandata service

Figure C summarises the added value of the ICAT service provided by the partners.



Server	Value	Deployment	Certificate	Content	Authentication
Alba – Spain					
DESY – Germany					
DLS – UK	50%	1	3	3	1
Elettra – Italy	0%	1	1	1	1
ESRF – France	50%	3	3	1	1
HZB – Germany	66%	3	3	1	2
ILL – France	66%	3	3	1	2
ISIS – UK	91%	3	3	3	2
LLB – France					
PSI – Switzerland					
Soleil – France	25%	3	1	1	1
STFC-UK	0%	1	1	1	1
Count		8	8	8	8
Average	43%	2.3	2.3	1.5	1.4
	score	Deployment characteristics			
	0	none	none	none	None
	1	other	self-signed	service-verfication	authn_db
	2	http:80	local	representative-data	institutional
	3	https:443	global	institutional	umbrella

Figure C: Added value

SV6 Conclusions

This service verification was successful and it was well supported by the partners.
The highlights include:

1. Production catalogs are now available;
2. Most of the partners can access data on the ICATs;
3. Some of the partners have successfully installed Nexus client software;
4. Most of the partners can access the Fuseki server;
5. All of the partners can access data in the European Affiliation Database;
6. The plan to provide a collection of good quality ICAT services is making good progress.