Deployment of Intersystems Caché with GUMS on Amazon EC2

prepared by: Daniel Tapiador
affiliation : ESAC Science Archives and VO Team
approved by: GAP
reference: GAIA-C9-TN-ESAC-DTP-002-0D
issue: 0D
revision: 0
date: 2011-10-18
status: Draft

Abstract

This technical note describes and analyzes the deployment of Intersystems Caché DBMS on the Cloud (Amazon EC2) with GUMS data.

Document History

Issue	Revision	Date	Author	Comment
D	0	2011-10-18	DTP	First draft.

Contents

1 Introduction **5**

1.1 Objectives 5

1.2 Reference Documents 5

1.3 Acronyms 5

2 Virtual Infrastructure **7**

2.1 Amazon EC2 7

2.2 Deployment with RightScale 11

2.3 I/O Performance Tuning 13

2.4 Conclusions 14

3 Intersystems Caché Deployment **15**

3.1 Configuration 15

3.2 DB, Namespace and Global Mappings 16

3.3 Ingestion and Further Optimizations 19

3.4 Scaling Out with ECP 21

3.5 Test Benchmarks 22

3.6 Conclusions 22

4 Intersystems DeepSee **23**

4.1 Data Cube definition and generation 23

4.2 GUMS Data Mining 24

4.3 Test Benchmarks 26

4.4 Conclusions 26

A EBS single and RAID volumes I/O benchmark on Amazon EC2 27

1 Introduction

This document describes the different steps taken to publish GUMS data through the commercial DBMS *Intersystems Caché* on *Amazon Elastic Compute Cloud (EC2)*. This deployment has been made with the purpose of benchmarking the performance obtained not only by the virtual infrastructure provided by the public cloud provider *Amazon EC2* but also by the DBMS itself as well as Intersystems DeepSee data mining framework. Furthermore, the problems encountered during this deployment are highlighted along with the solutions adopted.

1.1 Objectives

The Gaia catalogue will comprise more than one billion sources in its final delivery. Such amount of data cannot be easily dealt with by the traditional monolithic databases. Thus, distributed or parallel solutions will have to be tested and benchmarked not only for a better understanding of the data structure (multidimensional fields, etc) but also for a better knowledge of the expected usage by the community.

In this technical note, the high performance object database *Intersystems Caché* is tried out as well as the data mining product *Intersystems DeepSee*, with GUMS dataset. The configuration of these products and the results of the tests are shown together with the main conclusions reached, paying special attention to the scalability they offer and thus assessing the suitability for Gaia data.

In addition, the configuration, performance and pricing of the virtual infrastructure supplied by the Cloud vendor (Amazon) are also analyzed and compared to other public Cloud Computing providers. The Cloud Computing approach (both from a public and private cloud perspective) is also discussed for future reference.

1.2 Reference Documents

O'Mullane W., N.V., 2010, *Charting the Galaxy with the Gaia Satellite and InterSystems Caché*, Tech. rep., InterSystems and DPAC, `http://www.intersystems.com/cache/whitepapers/charting_the_galaxy.html`

1.3 Acronyms

The following is a complete list of the acronyms used in this document.

The following table has been generated from the on-line Gaia acronym list:

Acronym	Description
API	Application Programming Interface
AS	Intersystems Caché Application Server
AWS	Amazon Web Services
BI	Business Intelligence
CPU	Central Processing Unit
DAL	Data Access Layer
DB	DataBase
DBMS	DataBase Management System
DNS	Domain Name System
DPAC	Data Processing and Analysis Consortium
DS	Intersystems Caché Data Server
EBS	Elastic Block Store
ECP	InterSystems Enterprise Cache Protocol
EU	European Union
GAT	GOG Analysis Tool
GB	GigaByte
GUMS	Gaia Universe Model Snapshot
ID	Identifier (Identification)
IDE	Integrated Development Environment
IT	Information Technology
IaaS	Infrastructure as a Service
KB	KiloByte
MB	MegaByte
NAS	Network Attached Storage
OLAP	On-Line Analytical Processing
OO	Object Oriented
OS	Operating System
PaaS	Platform as a Service
RAID	Redundant Array of Inexpensive Disks
RHEL	Red Hat Enterprise Linux
SAN	Storage Area Network
SLA	Service Level Agreement
SQL	Structured Query Language
SSD	Solid State Disk
SSH	Secure SHell
SVN	SubVersioN
SaaS	Software as a Service
TB	TeraByte

TCP	Transmission Control Protocol
UI	User Interface
XEP	InterSystems Caché eXTreme Event Persistence

2 Virtual Infrastructure

The term Cloud Computing has many definitions, which can be summarized as a pool of configurable resources that can be provisioned and scaled on the fly. Then, it deals with the delivery of computing as a service rather than a product, like in the electricity grid, where users do not need to understand the component devices or infrastructure required to provide the service.

Furthermore, Cloud Computing encompasses three service models (all of them following the *pay as you go* pricing scheme):

- *Software as a Service (SaaS)*. The service provider develops the application as a set of interrelated components specified in the service manifest and defines the SLA that the infrastructure provider must comply with.
- *Platform as a Service (PaaS)*. It offers a development platform for developers. Applications can later on be deployed with one click.
- *Infrastructure as a Service (IaaS)*. Provision of processing power, storage, networks and other computing resources to the user.

At the moment of writing, there are quite a lot of Cloud Computing service providers. Some of them are *Amazon Web Services (AWS)*, *Google Apps*, *Microsoft Azure*, *Rackspace Cloud*, etc.

2.1 Amazon EC2

Amazon (AWS) has become one of the main actors of the cloud market, offering a wide range of cloud services such as Amazon Elastic Compute Cloud (EC2), Amazon Simple Storage Service (S3), Amazon Elastic MapReduce, etc. Amazon EC2 is the service chosen for deploying the resources within this exercise as it provides on-demmand access to computing power, and storage that can be seamlessly attached to virtual server instances. The way used to manage server instances within Amazon EC2 is described in more depth in section Sect. 2.2.

Amazon offers different geographically dispersed regions all around the world, being EU (Ireland) the one chosen for the tests due to its proximity. Within these regions, there are also some Availability Zones, which are distinct locations that are engineered to be insulated from failures

in other Availability Zones. It is important to take into account the Availability Zone where Elastic Block Store (EBS) disks are created as virtual servers will have to be deployed in the same Availability Zone for the attachment to succeed. In addition, one EBS disk can only be attached to one server. For mounting some storage in more than one instance, S3 must be used.

There are quite a lot of instance types offered by Amazon EC2, focused on both CPU or memory intensive applications as well as a few standard low-end instances. A quick look in the web will show all the details. The pricing for these instances is shown in Fig. 1. They correspond to *On-Demmand* instances that can be created and dismantled at any time. There is another category (Reserved Instances) where the customer makes one-time payment for the instance they want to reserve and in turn receive a significant discount on the hourly usage charge for that instance. After the one-time payment for an instance, that instance is reserved for the customer, and there is no further obligation; they may choose to run that instance for the discounted usage rate for the duration of the term. When the instance is not used, the customer will not be charged. This approach gives better prices (see Fig. 2) at a preliminary cost depending on the term period. This alternative might be worth considering for expected longer term deployments (not in this case). There is also a third type of instance (Spot Instances) which enables customers to bid for unused Amazon EC2 capacity. Instances are charged the Spot Price, which is set by Amazon EC2 and fluctuates periodically depending on the supply of and demand for Spot Instance capacity. Fig. 3 shows the lowest Spot Price per region and instance type (it is updated every 5 minutes so it shows a snapshot at the time of writing).

The instance chosen for the data server deployment is an On-Demmand *m2.2xlarge* with 34.2 GB of memory and 13 EC2 Compute Units which are distributed into 4 virtual cores in a 64-bit platform. Each Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

However, the total price to be charged will also include some other concepts such as the amount of Gigabytes of the different EBS and their snapshots as well as one metric dealing with the number of EBS I/O block hits (Fig. 4), and the data transferred from/to the Internet (shown in Fig. 5), or other Availability Zones (basically $0.01 per GB to other Availability Zones in the same Region). Furthermore, it is important to remark that although Amazon only charges for the actual time the virtual instance is up and running, the EBS disk space booked pricing metrics will apply even though they are not attached to any server. This is obvious as they have to keep that disk space reserved with permanent user data on it, but worth mentioning.

With regard to I/O performance, there are some I/O performance indicators for each type of instance, being *low*, *moderate* and *high*. The instance selected for holding the main data server in this deployment has high I/O performance which is likely to correspond to 1 Gigabit Ethernet although this is something not confirmed by Amazon (thus around 120 MB/s is the top transfer rate achievable). There is another type of I/O performance indicator (*very high*) which is thought to be used by clusters and which provides 10 Gibagit Ethernet with reduced network latencies within the nodes of the cluster.

Region: EU (Ireland)

	Linux/UNIX Usage	Windows Usage
Standard On-Demand Instances		
Small (Default)	$0.095 per hour	$0.12 per hour
Large	$0.38 per hour	$0.48 per hour
Extra Large	$0.76 per hour	$0.96 per hour
Micro On-Demand Instances		
Micro	$0.025 per hour	$0.035 per hour
Hi-Memory On-Demand Instances		
Extra Large	$0.57 per hour	$0.62 per hour
Double Extra Large	$1.14 per hour	$1.24 per hour
Quadruple Extra Large	$2.28 per hour	$2.48 per hour
Hi-CPU On-Demand Instances		
Medium	$0.19 per hour	$0.29 per hour
Extra Large	$0.76 per hour	$1.16 per hour
Cluster Compute Instances		
Quadruple Extra Large	N/A*	N/A*
Cluster GPU Instances		
Quadruple Extra Large	N/A*	N/A*
* Cluster Compute and Cluster GPU Instances are currently only available in the US East (Virginia) Region.		

FIGURE 1: Amazon EC2 On-Demmand instance pricing for EU (Ireland)

Region: EU (Ireland)

	1 yr Term	3 yr Term	Linux/UNIX Usage	Windows Usage
Standard Reserved Instances				
Small (Default)	$227.50	$350	$0.04 per hour	$0.06 per hour
Large	$910	$1400	$0.16 per hour	$0.24 per hour
Extra Large	$1820	$2800	$0.32 per hour	$0.48 per hour
Micro Reserved Instances				
Micro	$54	$82	$0.01 per hour	$0.016 per hour
High-Memory Reserved Instances				
Extra Large	$1325	$2000	$0.24 per hour	$0.32 per hour
Double Extra Large	$2650	$4000	$0.48 per hour	$0.64 per hour
Quadruple Extra Large	$5300	$8000	$0.96 per hour	$1.28 per hour
High-CPU Reserved Instances				
Medium	$455	$700	$0.08 per hour	$0.145 per hour
Extra Large	$1820	$2800	$0.32 per hour	$0.58 per hour
Cluster Compute Reserved Instances				
Quadruple Extra Large	N/A*	N/A*	N/A*	N/A*
Cluster GPU Reserved Instances				
Quadruple Extra Large	N/A*	N/A*	N/A*	N/A*
* Cluster Compute and Cluster GPU Instances are currently only available in the US East (Virginia) Region.				

FIGURE 2: Amazon EC2 Reserved instance pricing for EU (Ireland)

Region: EU (Ireland)

	Linux/UNIX Usage	Windows Usage
Standard Spot Instances		
Small (Default)	$0.04 per hour	$0.067 per hour
Large	$0.16 per hour	$0.258 per hour
Extra Large	$0.32 per hour	$0.533 per hour
Micro Spot Instances		
Micro	$0.01 per hour	$0.016 per hour
High-Memory Spot Instances		
Extra Large	$0.24 per hour	$0.32 per hour
Double Extra Large	$0.56 per hour	$0.733 per hour
Quadruple Extra Large	$1.12 per hour	$1.467 per hour
High-CPU Spot Instances		
Medium	$0.08 per hour	$0.167 per hour
Extra Large	$0.32 per hour	$0.667 per hour
Cluster Compute Instances		
Quadruple Extra Large	N/A*	N/A*
Cluster GPU Instances		
Quadruple Extra Large	N/A*	N/A*
* Cluster Compute and Cluster GPU Instances are currently only available in the US East (Virginia) Region.		

FIGURE 3: Amazon EC2 Spot instance pricing for EU (Ireland)

Region: EU (Ireland)

Amazon EBS Volumes

- $0.11 per GB-month of provisioned storage
- $0.11 per 1 million I/O requests

Amazon EBS Snapshots to Amazon S3

- $0.15 per GB-month of data stored
- $0.01 per 1,000 PUT requests (when saving a snapshot)
- $0.01 per 10,000 GET requests (when loading a snapshot)

FIGURE 4: Amazon EC2 pricing per GB of EBS and per million I/O requests for EU (Ireland)

Region: EU (Ireland)

	Pricing
Data Transfer IN	
All data transfer in	$0.000 per GB
Data Transfer OUT	
First 1 GB / month	$0.000 per GB
Up to 10 TB / month	$0.120 per GB
Next 40 TB / month	$0.090 per GB
Next 100 TB / month	$0.070 per GB
Next 350 TB / month	$0.050 per GB
Next 524 TB / month	Contact Us
Next 4 PB / month	Contact Us
Greater than 5 PB / month	Contact Us

FIGURE 5: Amazon EC2 data transfer from/to Internet pricing for EU (Ireland)

2.2 Deployment with RightScale

Rightscale is a cloud management platform that allows users to perform complete application deployments, comprising multiple servers and connections between them, or across one or more clouds, managing as well the resource contextualization through templates and scripts (RightScripts). All this done through a nice web User Interface (UI) that hides the sometimes complex and low-level AWS syntax.

Although there is not any script ready for Intersystems Caché yet and thus it has had to be installed manually as further explained in 3, it is worth stating that the recommended way to do this is through the usage of RightScripts so that the installation can be automated for future deployments of i.e. Intersystems Caché Application Servers (AS) or any other software module that needs to be deployed in the server instances. Furthermore, the OS chosen from the templates available in RightScale has been CentOS 5.6 (x86_64) as Red Hat Enterprise Linux (RHEL) is not available yet through templates due to licensing issues.

Fig. 6 shows one screenshot of the UI for managing a server instance. There are some controls for cloning the server, stopping, rebooting, etc. and also other tabs for configuring the EBS disks attached, monitoring and so on. Security is handled by using security groups which are basically firewall like entries for allowing or banning determined network traffic.

For connecting to the host as the administrator (*root*) through SSH, apart from opening the corresponding TCP port in the security group, the public DNS address and the private SSH key supplied must be used (both are easily accessible in RightScale web pages for each server instance).

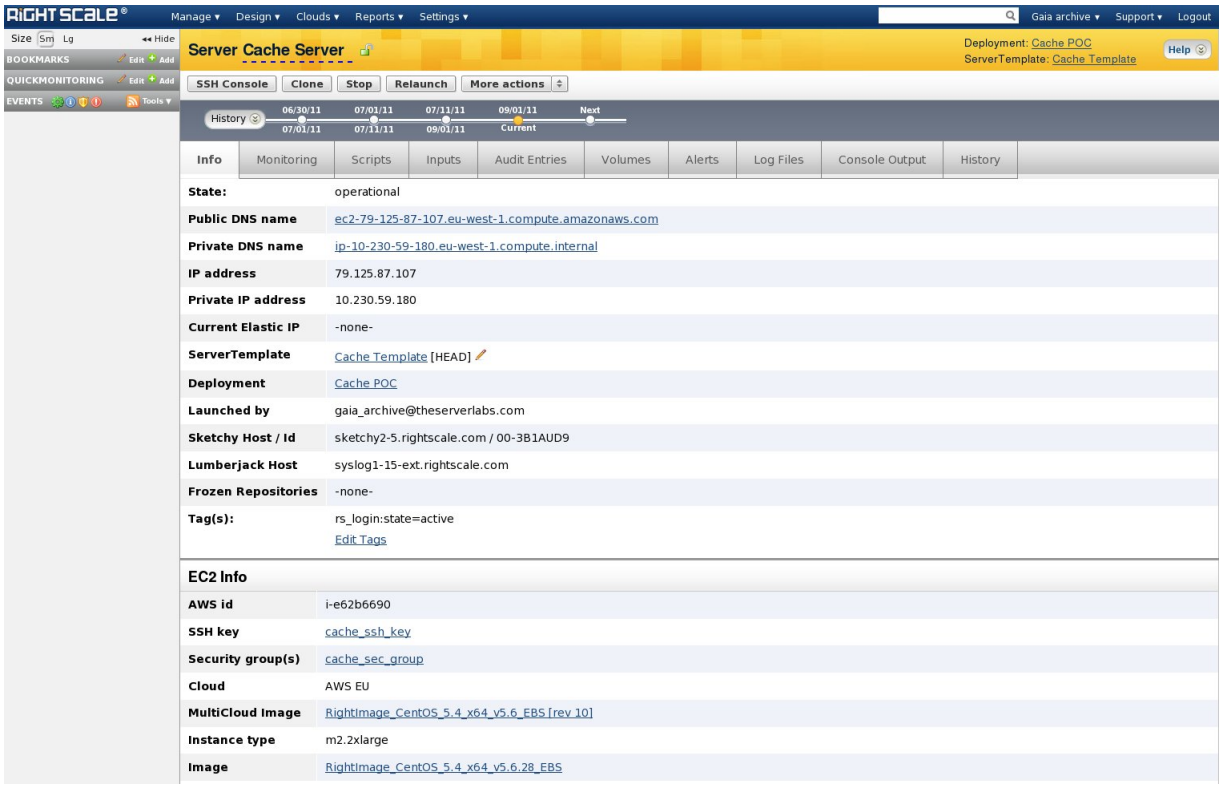


FIGURE 6: RightScale screenshot for a server deployed in the Amazon EC2

2.3 I/O Performance Tuning

As already discussed in Sect. 2.1, most of the components for the different server instances can be clearly specified (number of CPUs, amount of memory and disk, CPU architecture, etc), although there are some metrics (i.e. I/O performance) that are loose due to the inherent constraints of Cloud Computing approachh. In the particular case of this deployment, I/O performance is one of the key aspects for the success of the experiment, overall when dealing with such amount of data (GUMS dataset).

Considering that EBS disks are mounted over the network and, as in any other Cloud deployment, Amazon instances are likely to share the network bandwidth available with other server instances (even belonging to other customers), it is important to analyse the different configurations available for both getting the maximum performance possible and minimizing the impact of the bandwidth sharing among the different instances deployed within the same resources our server runs on, thus sustaining the I/O performance obtained over time no matter how much use of I/O resources the other instances are making.

There are quite a lot of studies in this respect on the Internet (one of them shown in Apndx. A), giving very accurate numbers of the different disk configurations. As expected, those grouping different disks like RAID0, RAID5 and RAID10 give the best performance, although there are some peculiarities of each one. For instance, RAID10 and RAID5 give some kind of redundancy which we do not need at all for this exercise as the data can be ingested again in case of a disaster. Then, the most reasonable approach is to use RAID0 (block striping) for maximizing the amount of disk devoted to GUMS data storage (no redundancy at all through parity or replication of blocks).

The next questions to be answered refer to the number of disks to be used when using RAID0 configuration, and when this RAID0 should be chosen and when other configurations (even single EBS disks) might be set up.

To answer the first question, we need to know how the database works. Ideally, when running a query over some columns with a set of constraints, there will be indices for the fields in the constraint part (*where* clause) and thus it will be known what rows to retrieve from the disk (which are likely to be spread over it). In this typical scenario, a good random read access is needed to the disk so that the maximum number of rows can be retrieved per unit of time. The table shown in Apndx. A states that the best performance for random reads is RAID0 (with four disks) configuration mainly due to the fact that when one data block is being retrieved from one disk, the heads of the other disks can be moving to where the next data blocks are (instead of waiting for the first data block reading operation to complete which would happen if next blocks resided on the same physical disk).

Having a look at the numbers shown for sequential reads, there is not much performance gain when using multiple disk RAID configurations comparing to the single EBS disk as the limit

of the available instance network bandwidth may already be reached with one EBS volume for streaming workloads. Therefore, considering that indices are written and read sequentially on the disk (*ordered index*), a single EBS has been used for holding GUMS data indices as further explained in Sect. 3.

2.4 Conclusions

Amazon Web Services (AWS) has become the *de facto* standard for Cloud Computing services (Amazon EC2 for IaaS, Elastic MapReduce, etc) as among other things, the infrastructure may be considered operational and they achieve a very good level of service, allowing companies to scale up on demmand or to fully port their applications to the Cloud.

However, Amazon procedures for failure and disaster recovery are not yet at the level one might expect as it has been proven in the last incident suffered in one of the availability zones in the EU data centre (in Ireland) on August 7th, 2011, when a lightning strike caused a transformer to fail and the backup generators did not start, causing almost all EC2 instances and about 58% of the EBS volumes (in that availability zone, not the entire region) to go down.

The main problem for our own deployment was that when the instance was brought back to life again, some of the EBS were not connected to it, preventing the services running on the server from resuming operations normally. Actually, the EBS containing GUMS database indices was completely gone and thus they have had to be fully recreated again. The other EBS disks could be recovered manually from the automatic snapshots taken so the incident was not so hard to recover from in the end.

As Amazon continues to evolve their systems and sort out any other possible issues that may occur in situations like the one explained above, there are also some tasks that should be addressed by the customers themselves to prevent this kind of downtimes from happening again by, for instance, replicating core services, performing regular snapshots and keeping them permanently stored in other availability zones (or even regions), etc.

The future perspectives for Cloud Computing are good as this technology is already being used by many companies, overall small companies, which are being the early adopters (not the other way around as it has always happened for traditional IT innovations). Furthermore, the number of customers making use of private and public clouds will grow more and more over this decade as this model gives more flexibility and scalability, reducing to practically zero the fixed costs for computing power.

3 Intersystems Caché Deployment

Intersystems Caché is a commercial high performance object database aimed at handling huge volumes of transactional data. It provides several different ways of accessing the information (relational with SQL, Object-Oriented, etc) with APIs for several programming languages including Java or C++ among others. Intersystems also offers an ecosystem around the product itself with interesting tools like DeepSee (for data mining) and iKnow (for natural language processing).

One of the key aspects of this database is the possibility to seamlessly deal with multidimensional data fields, hiding the complexity of arrays manipulation, maps, object references and any other nested structures that are normally present in the OO world.

In addition, the database can work in a distributed environment by making use of the powerful Enterprise Cache Protocol (ECP). ECP is a high performance and scalable technology that enables computers in a distributed system to use each other's databases. The use of ECP requires no application changes (applications simply treat the database as if it were local).

In this section, the different configuration options as well as the steps and optimizations carried out for deploying GUMS in the database will be discussed along with some test benchmarks run over the database product running in Amazon EC2.

3.1 Configuration

Intersystems Caché can be easily installed in Linux by uncompressing the distribution kit and running the self-guided installation script. Please refer to the documentation for further details at `http://docs.intersystems.com/cache.html`. The version of the package chosen for these tests is 2011.1.0.474 and the Linux flavour has been Red Hat Enterprise Linux 5 (x86_64 architecture) although the real OS deployed in Amazon EC2 is CentOS 5 (x86_64) as RHEL is not yet available as a RightScale script due to licensing issues and CentOS is the very same thing as RHEL. Actually, CentOS is the compilation of the source code released by Red Hat Enterprise Linux.

The main configuration setting that has to be changed is the memory allocated to the database engine. As previously stated in Sect. 2.1, the virtual instance chosen for running the database server has around 34 GB of main memory, then it is recommended to devote around half of it to Caché database (16 GB).

Journaling is a powerful feature for databases. It records every operation applied to the database so that in case of a hardware fault or a network failure the database system can restore the data to the original state. In our case, this feature has been disabled for gaining performance in the ingestion process. The reason why journaling is disabled comes from the fact that all the

data that need to be ingested in the system (GUMS) are already stored in the original binary and compressed (gbin format) files delivered. Then, no data loss may occur. The easiest way to disable journaling in the database is as follows (it must be done whenever the database is restarted as by default it starts with it enabled):

OS SHELL>**csession ds**
USER>**zn "%SYS"**
%SYS>**do ^JOURNAL**
(choose option 2)

A separated EBS has been devoted to *CACHE.WIJ* in order to get a better performance for this write image journal file access, although it should ideally go to a (much faster) Solid State Disk (SSD) or the like.

The rest of the configuration will be discussed in the pertinent subsections below.

3.2 DB, Namespace and Global Mappings

Intersystems Caché performance is speeded up by adding different disks to the data server and partitioning the data among them. It is mandatory to highlight that it is not a pure parallel database where data are physically stored in the different nodes of the cluster and they all work in close collaboration for executing queries through complex parallel algorithms (for parallel sort, etc). Intersystems Caché is more a distributed database where parallelism comes from another layer of nodes (called *Application Servers (AS)*) that cache data for the different clients which is always retrieved from the *Data Server (DS)* (the one whose configuration is being explained in this section).

There is another powerful feature that allows to improve performance in the DB. This feature is the partitioning of data among different disks for parallel I/O, so that different objects are spread over different disks. This partitioning is managed by the DB engine itself although the administrator must defined what data go to what particular disk. As it is shown below, this is called *Global Mappings* in the Intersystems Caché technical jargon.

Therefore, a set of 4 disks have been created (each disk is formed by a RAID0 with 4 EBS of as explained in Sect. 2.3). Three of the disks have been physically assembled by using 4 EBS of 110 GB each so that the server sees 3 physicall disks of around 430 GB. The other disk is formed by 4 EBS of 365 GB, obtaining a disk of around 1.5 TB, as this disk will contain the information for DeepSee tests (apart from its slice of GUMS data).

Once the disks have been attached and mounted in the server, 4 local databases have been created with journal disabled and 8 KB block size. Once a local database is created, Intersystems Caché creates a file called *CACHE.DAT* which will contain the user data. In order to improve

performance when ingesting GUMS, the file should have the maximum capacity that it will use already allocated to it (not to lose time while ingesting by enlarging the file capacity every time it gets full). Therefore, this file size must be estimated depending on the data it will hold, and will have to be created with that size allocated before writing user data to it. Then, three of the disks will have *CACHE.DAT* files of around 430 GB and the other one will have a file with 1.5 TB already allocated. The file size can be specified within the wizard that creates the DB, but it will take longer to create it in that way. As an alternative way, the fastest way of allocating the space on those files is by creating only one DB with the required size by using the wizard (the rest of databases have to be created but with 1 MB as their size only) and then stopping the database server and overwriting the 1 MB files with the one with the estimated size (430 GB in this case for the 3 local DB on the 430 GB disks). This copy process (running at the OS level) will take less time. Once the files have been copied, the database can be started again and it will be ready for ingestion.

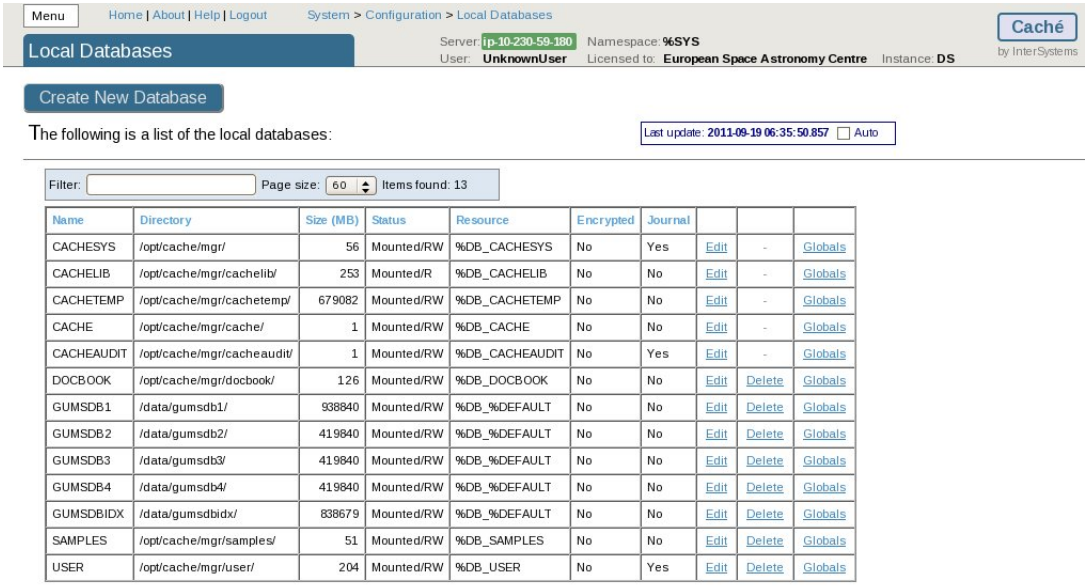
This procedure is summarized below:

- Create one local DB with the estimated disk space already pre-allocated.
- Create the rest of local DBs with the default space (1 MB).
- Shutdown Caché.
- Overwrite *CACHE.DAT* of the local DB with the default space (1 MB) with the larger *CACHE.DAT* created in the first step.
- Start up Caché.

Fig. 7 shows the different DBs created for GUMS (GUMSDB1 to GUMSDB4 and GUMSDBIDX).

Once the DBs have been created with the proper disk space pre-allocated, it is time to create the namespace. The namespace is something that can be used among other things to group several different local databases so that it appears as only one (even though it is composed of different local DBs stored internally in different disks). Fig. 8 shows the namespace (GUMSNS) used for gathering the different local databases and for giving a uniform access to the GUMS data. The local DB used by default for holding *globals* and *routines* as well as the one for storing temporary data must be specified.

Intersystems Caché stores data into what they call *globals*. There is a different *global* for any data object type that is stored in the DB. If the *globals* are not mapped, they will be stored in the default DB for *globals* specified in the corresponding namespace (See Fig. 8). For distributing data among the different local DBs (thus in the different disks), the global mapping section must be accessed. Fig. 9 shows the mappings done for the GUMS type UMStellarSource. The *global*


Menu Home | About | Help | Logout System > Configuration > Local Databases

Local Databases

Server: ip-10-230-59-180 Namespace: %SYS
User: UnknownUser Licensed to: European Space Astronomy Centre Instance: DS

Caché by InterSystems

Create New Database

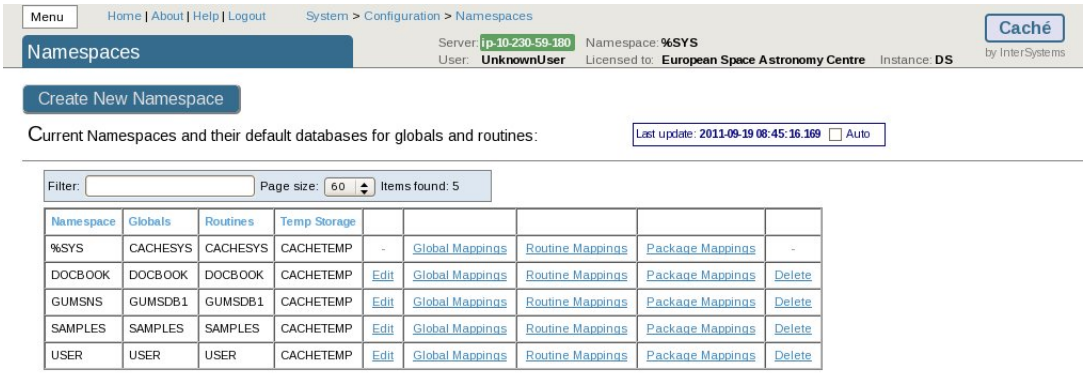
The following is a list of the local databases:

Last update: 2011-09-19 06:35:50.857 Auto

Filter: Page size: 60 Items found: 13

Name	Directory	Size (MB)	Status	Resource	Encrypted	Journal			
CACHESYS	/opt/cache/mgr/	56	Mounted/RW	%DB_CACHESYS	No	Yes	Edit	-	Globals
CACHELIB	/opt/cache/mgr/cachelib/	253	Mounted/R	%DB_CACHELIB	No	No	Edit	-	Globals
CACHETEMP	/opt/cache/mgr/cachetemp/	679082	Mounted/RW	%DB_CACHETEMP	No	No	Edit	-	Globals
CACHE	/opt/cache/mgr/cache/	1	Mounted/RW	%DB_CACHE	No	No	Edit	-	Globals
CACHEAUDIT	/opt/cache/mgr/cacheaudit/	1	Mounted/RW	%DB_CACHEAUDIT	No	Yes	Edit	-	Globals
DOCBOOK	/opt/cache/mgr/docbook/	126	Mounted/RW	%DB_DOCBOOK	No	No	Edit	Delete	Globals
GUMSDB1	/data/gumsdb1/	938840	Mounted/RW	%DB_%DEFAULT	No	No	Edit	Delete	Globals
GUMSDB2	/data/gumsdb2/	419840	Mounted/RW	%DB_%DEFAULT	No	No	Edit	Delete	Globals
GUMSDB3	/data/gumsdb3/	419840	Mounted/RW	%DB_%DEFAULT	No	No	Edit	Delete	Globals
GUMSDB4	/data/gumsdb4/	419840	Mounted/RW	%DB_%DEFAULT	No	No	Edit	Delete	Globals
GUMSDBIDX	/data/gumsdbidx/	838679	Mounted/RW	%DB_%DEFAULT	No	No	Edit	Delete	Globals
SAMPLES	/opt/cache/mgr/samples/	51	Mounted/RW	%DB_SAMPLES	No	No	Edit	Delete	Globals
USER	/opt/cache/mgr/user/	204	Mounted/RW	%DB_USER	No	Yes	Edit	Delete	Globals

FIGURE 7: Local DBs and their attributes. The ones used for GUMS are GUMSDB1 to GUMSDB4 and GUMSDBIDX.


Menu Home | About | Help | Logout System > Configuration > Namespaces

Namespaces

Server: ip-10-230-59-180 Namespace: %SYS
User: UnknownUser Licensed to: European Space Astronomy Centre Instance: DS

Caché by InterSystems

Create New Namespace

Current Namespaces and their default databases for globals and routines:

Last update: 2011-09-19 08:45:16.169 Auto

Filter: Page size: 60 Items found: 5

Namespace	Globals	Routines	Temp Storage					
%SYS	CACHESYS	CACHESYS	CACHETEMP	-	Global Mappings	Routine Mappings	Package Mappings	-
DOCBOOK	DOCBOOK	DOCBOOK	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete
GUMSNS	GUMSDB1	GUMSDB1	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete
SAMPLES	SAMPLES	SAMPLES	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete
USER	USER	USER	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete

FIGURE 8: Namespaces. GUMSNS is the one aggregating local DBs that contain GUMS data.

name is based on the Java package name and the class name cut off with some random code at the end for abbreviation. The last letter shows whether we are referring to data (*D*) or indices (*I*). This way, all indices will go to GUMSDBIDX local database whose disk is formed by a single EBS as indices are often read sequentially and there is no significant performance improvement from single to RAID configurations when reading is done sequentially. The subscript defines the amount of rows that will go to the different local DBs. In this case, the first 710,000,000 rows will go to *GUMSDB1* local DB, rows whose ID is from 710,000,001 to 1,420,000,000 will be placed in *GUMSDB2*, and so on. This ID is generated internally by Caché for every row as they are being ingested so that the first object inserted will have the ID 1 and so on. Ideally for the Gaia catalogue, it would be very convenient to have the objects representing sources ordered by their positions in the sky, so that each local DB contains the data for a well defined region in the sky. This can be achieved by computing the Healpix ID for each source and then pre-order the data by that identifier. Once the data objects are ordered, ingestion will be performed following that order. For the tests explained in this document, no data ordering has been performed prior to ingestion (objects have been inserted in the DB unordered) as the different strategies for doing so are, at the time of writing, being further explored.

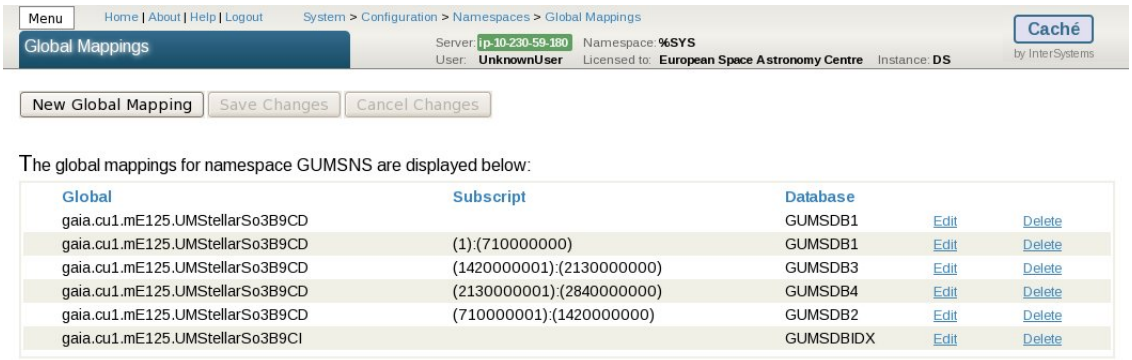


FIGURE 9: Global mappings for GUMSNS namespace.

3.3 Ingestion and Further Optimizations

As previously stated in Sect. 3, Intersystems Caché provides different ways of accessing and manipulating the data, one of it being the Java API called XEP (eXTreme Event Persistence). XEP allows for projection of simple Java objects as persistent events. It is basically a thin layer of software that implements a simple object to global mapping which provides high rates of object storage and retrieval. Furthermore, XEP eliminates the object/relational mapping thus shortening development time dramatically and boosting runtime performance. The drawback is that some flexibility is lost as XEP is not a standard for accessing datasets. The question to answer in this respect is whether flexibility is needed more than performance or the other way around.

The way XEP has been used for ingesting GUMS data is through GaiaTools DAL module for In-

tersystems Caché. More details over this DAL module as well as required attributes and configuration options can be found in DPAC SVN at `http://gaia.esac.esa.int/dpacsvn/DPAC/CU1/docs/TechNotes/CacheTechNote/`. The Gaia DAL abstraction is highly configurable and very easy to use, resulting in clean and clear ingestion source code.

Several different architectures have been tried out for GUMS ingestion in the DS running at Amazon EC2. The one finally used is the most naive one, running a sequential process in the DS (which gave around 35,000 to 40,000 objects per second), as it offered the best performance due to mainly I/O bottlenecks in Amazon EC2 infrastructure for other parallel configurations using ECP with multiple AS servers ingesting in parallel through the powerful DS server (which gave around 48,000 objects per second but not sustained over time). However, ECP has proven to be a very good feature for ingestion as stated in O'Mullane W. (2010) (with 112,000,000 objects per second), so it would be expected that performance obtained in Amazon EC2 were substantially increased if powerful I/O technologies such as 10 Gigabit Ethernet or Fibre Channel to disks were used. In addition, when running on a public cloud infrastructure like Amazon EC2, there will always be constraints with regard to the control of peaks of I/O operations produced by other third-party customers running their applications. As a conclusion, the monitoring of the system resources consumption while ingesting GUMS into the DB, it can be clearly spotted that the bottleneck is the I/O channels as the CPU is most of the time waiting for I/O operations to complete. Nonetheless, ingestion is around 10 times faster than in other DBMS like PostgreSQL and could be much improved in an operational deployment with powerful SAN or NAS, and 10 Gigabit Ethernet as stated above.

Another optimization done for improving query performance is the table tuning for calculating the selectivity and extentsize. This is very useful for queries with multiple constraints as these constraints will be reordered so as to reduce the number of objects to check against, etc.

In addition to previous optimizations, some indices have been created for speeding up query execution. Fig. 10 shows the indices names and the corresponding columns they are indexing. Most of the indices are normal traditional indices, but a few of them are bitmap indices which are ideal for exact match queries. There is another type of index named bitslice which are ideal for range queries, although they have not been used as GUMS UMStellarSource fields are highly dispersed in general. It is important to remark that Caché needs a huge temporal storage space as expected and that should be taken into account before starting the indices generation background job.

The fastest way to create indices is through the Studio IDE. The steps are shortly summarized below:

- Create one index by using the wizard in the Studio IDE.
- Open the class definition file and add the other indices (based on the syntax for the index created in the first step) by using the editor.

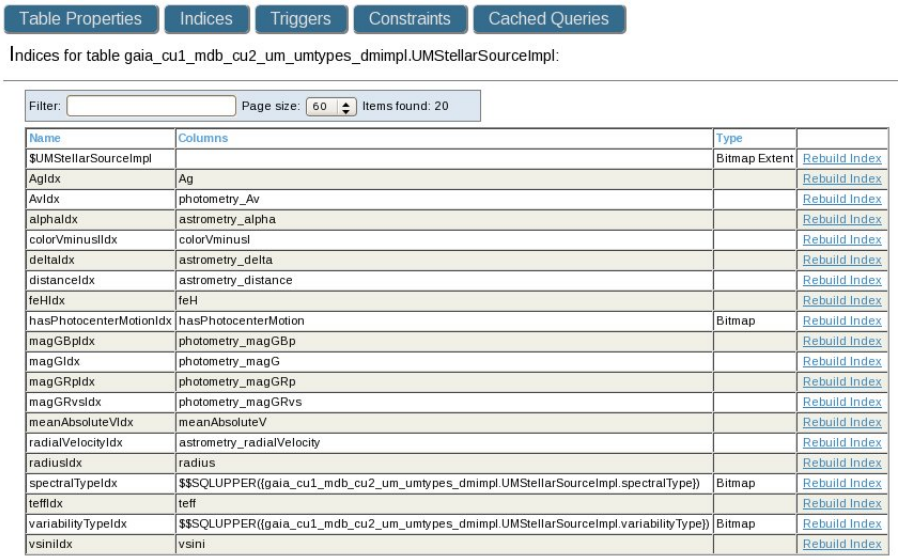


Table Properties | Indices | Triggers | Constraints | Cached Queries

Indices for table gaia_cu1_mdb_cu2_um_umtypes_dmimpl.UMStellarSourceImpl:

Filter: Page size: 60 Items found: 20

Name	Columns	Type	
$UMStellarSourceImpl		Bitmap Extent	Rebuild Index
AgIdx	Ag		Rebuild Index
AvIdx	photometry_Av		Rebuild Index
alphaIdx	astrometry_alpha		Rebuild Index
colorVminusIIdx	colorVminusI		Rebuild Index
deltaIdx	astrometry_delta		Rebuild Index
distanceIdx	astrometry_distance		Rebuild Index
feHIdx	feH		Rebuild Index
hasPhotocenterMotionIdx	hasPhotocenterMotion	Bitmap	Rebuild Index
magGBpIdx	photometry_magGBp		Rebuild Index
magGIdx	photometry_magG		Rebuild Index
magGRpIdx	photometry_magGRp		Rebuild Index
magGRvsIdx	photometry_magGRvs		Rebuild Index
meanAbsoluteVIdx	meanAbsoluteV		Rebuild Index
radialVelocityIdx	astrometry_radialVelocity		Rebuild Index
radiusIdx	radius		Rebuild Index
spectralTypeIdx	$$SQLUPPER({gaia_cu1_mdb_cu2_um_umtypes_dmimpl.UMStellarSourceImpl.spectralType})	Bitmap	Rebuild Index
teffIdx	teff		Rebuild Index
variabilityTypeIdx	$$SQLUPPER({gaia_cu1_mdb_cu2_um_umtypes_dmimpl.UMStellarSourceImpl.variabilityType})	Bitmap	Rebuild Index
vsiniIdx	vsini		Rebuild Index

FIGURE 10: Indices created for UMStellarSource objects.

- Compile the class through the Studio IDE or the web portal.
- Start indices generation process in the background through the web portal (System, SQL, Schemas, Tables, Our Table, Rebuild Indices).

3.4 Scaling Out with ECP

As taken from InterSystems web pages, *InterSystems' Enterprise Cache Protocol (ECP) is designed to dramatically enhance the scalability and performance of distributed applications. Optimized for thin-client architectures, ECP makes network traffic between application servers and the database more efficient, thus allowing the network to support an expanded middle tier and more users*.

Then, the ideal architecture would comprise a powerful I/O Data Server (DS) with several middle tier Application Servers (AS) retrieving data from the DS in an efficient manner by making a very thorough usage of the cache memory of AS with the advanced mechanisms included in ECP. The users will therefore connect to the AS (always to the same one for taking advantage of cache hits), allowing the number of users to be significantly increased, which will run on thin architectures like laptops or even other devices like tablets or smart phones.

For configuring ECP (see Fig. 11), the DS must enable the service and configure a few parameters on the right side (*This System as an ECP Data Server*). Later on, in the AS, the parameters shown on the left side (*This System as an ECP Application Server*) must be configured, and the remote DS this AS connects to must be added through the button at the bottom.

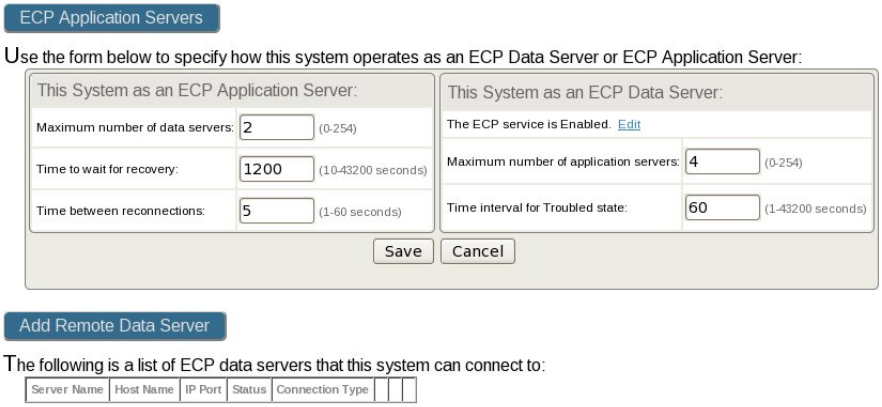


FIGURE 11: Configuration page for ECP.

3.5 Test Benchmarks

Benchmark for test queries executed against GUMS dataset.

3.6 Conclusions

Although the deployment of Intersystems Caché on Amazon EC2 has been successful, there are some aspects that should be thoroughly addressed before studying the possibility of deploying a more permanent solution on a public cloud infrastructure. The main issue would be to figure out how to improve I/O performance to acceptable levels for the vast amount of data produced by the Gaia mission. This might be accomplished by deploying other *cluster* configurations offered by Amazon EC2 which claim to provide 10 Gigabit Ethernet connections.

For optimizing spatial queries, it might also be good to ingest sources ordered by Healpix indices so that sources having coordinates near to each other will be placed physically next to the other on the disk for improving retrieval performance. For that, different alternatives (like MapReduce) will have to be studied so that data coming from the mission can be efficiently ordered before going on with the ingestion.

The ingestion itself should be done in parallel by using several AS and splitting the data in different well balanced partitions with the *global mappings* capabilities offered by Caché, and setting *ecpRangeSize* property later on for properly assigning the available IDs to each AS.

More conclusions about the benchmark for test queries run against GUMS dataset.

4 Intersystems DeepSee

InterSystems DeepSee is a data mining tool that enables users to embed business intelligence (BI) into their applications so that they can ask and answer sophisticated questions of their data.

One of the most interesting use cases for Gaia data will be the creation of different kinds of statistics like the number of objects per type, the sky density for Healpix areas (at different granularities), the count per spectral type and magnitude range or any other combination of dimensions and measurements available. Ideally, it would be desirable that these charts are computed on the fly so that users are given the flexibility they need for analyzing the Gaia data without having to precompute the statistics in advance.

Intersystems DeepSee allows this capability of manipulating and analyzing data from multiple perspectives. For that, an On-Line Analytical Processing (OLAP) cube must be built. This cube, also known as *multidimensional cube* or *hypercube*, can be thought of as extensions to the two-dimensional array of a spreadsheet. The most common example of a data cube is the one used by companies which want to analyze some financial data by product, time period, city, type of revenue and cost, or by comparing actual data with a budget.

The OLAP cube consists of numerical facts called measures (sales volume, revenue, etc) which are categorized by dimensions (product type, date range, geographical region, etc). In the particular case of Gaia data, counts or fields like effective temperature or radial velocity might be considered as measures, and position (Healpix ID or alpha and delta ranges), magnitude or spectral type might be taken as dimensions, although it might be the other way around, as it always depends on the knowledge to be extracted from the data.

DeepSee data mining portal comprises several tools which can be used for creating data models (DeepSee Architect), pivot tables (DeepSee Analyzer), web portals based on the pivot tables, etc. The following subsections describe these components in more detail.

4.1 Data Cube definition and generation

Fig. 12 shows the main window of Intersystems DeepSee Architect. It is the tool to define cubes that will be later on accessed through DeepSee Analyzer. It basically contains one vertical panel on the left where we can select fields for drag and drop to other places. The vertical panel on the middle is where we drop fields, each into the sections for Measures, Dimensions, Listings, etc. The panel on the right shows information about the selected field (already present in one of the sections for Measures, Dimensions, etc). Once the cube is defined and saved, it can be compiled and built. The building process may take long if the dataset the cube is generated from is large, like in this case (GUMS UMStellarSource entity contains around 2.6 billion objects). Just out of curiosity, DeepSee has taken approximately 2.5 days to build the test cube described below.

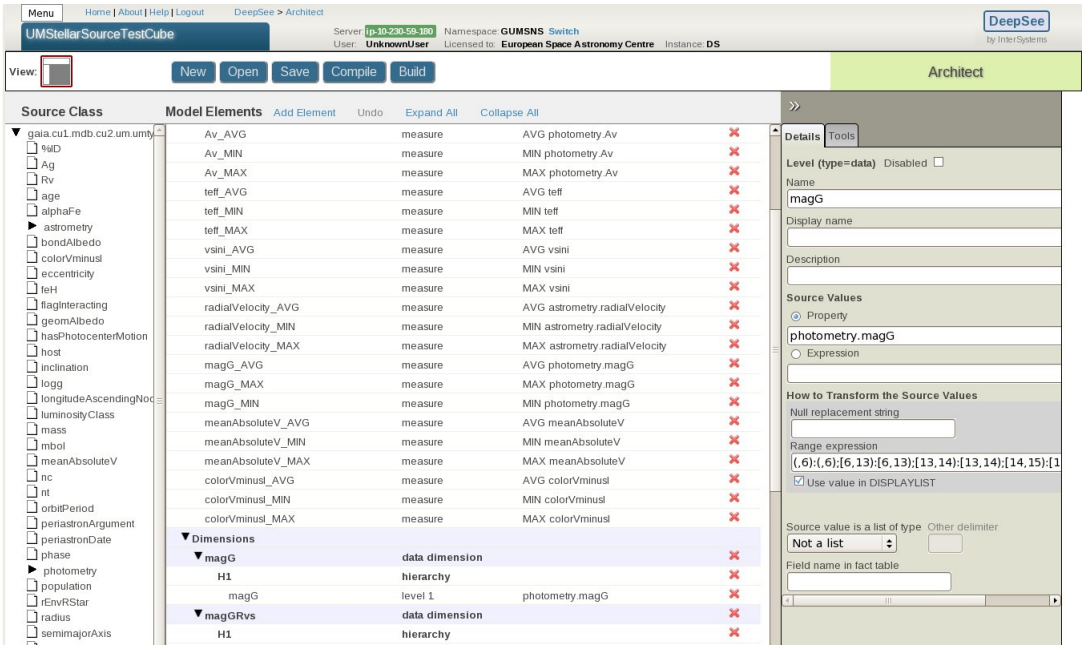


FIGURE 12: DeepSee Architect main view.

UMStellarSourceTestCube is a test cube that has been built for testing purposes. It takes the Gaia Science Performance (`http://www.rssd.esa.int/index.php?project=GAIA&page=Science_Performance`) as inputs for dimensions, measures, etc. A careful analysis has been made throughout the web page for gathering the different interesting dimensions for astrometry, photometry, etc. Furthermore, the ranges for those dimensions have been taken from the values written on those tables. Other fields containing a relatively small set of values (spectralType, hasPhotocenterMotion, etc) have also been included as dimensions (without specifying any range as it is not needed due to the discrete set of values they contain). Counts have not been included in the measures section as DeepSee does so by default.

4.2 GUMS Data Mining

DeepSee Analyzer allows to create and modify pivot tables on the fly for the data models (cubes) already created through DeepSee Architect. Fig. 13 shows the DeepSee Analyzer main view for the test data cube built in Sect. 4.1 (UMStellarSourceTestCube).

As it can be seen in Fig. 13, we can drag and drop dimensions over the *Rows* and *Columns* panels to build the pivot tables we may be interested in. In this particular case, the pivot table built shows the number of sources per *magG* and *colorVminusI* categories, but any other combination of dimensions (even in a hierarchical fashion) and measures can also be done. Filters can also be added to restrict the number of cells to build. It is important to remark that for a dataset

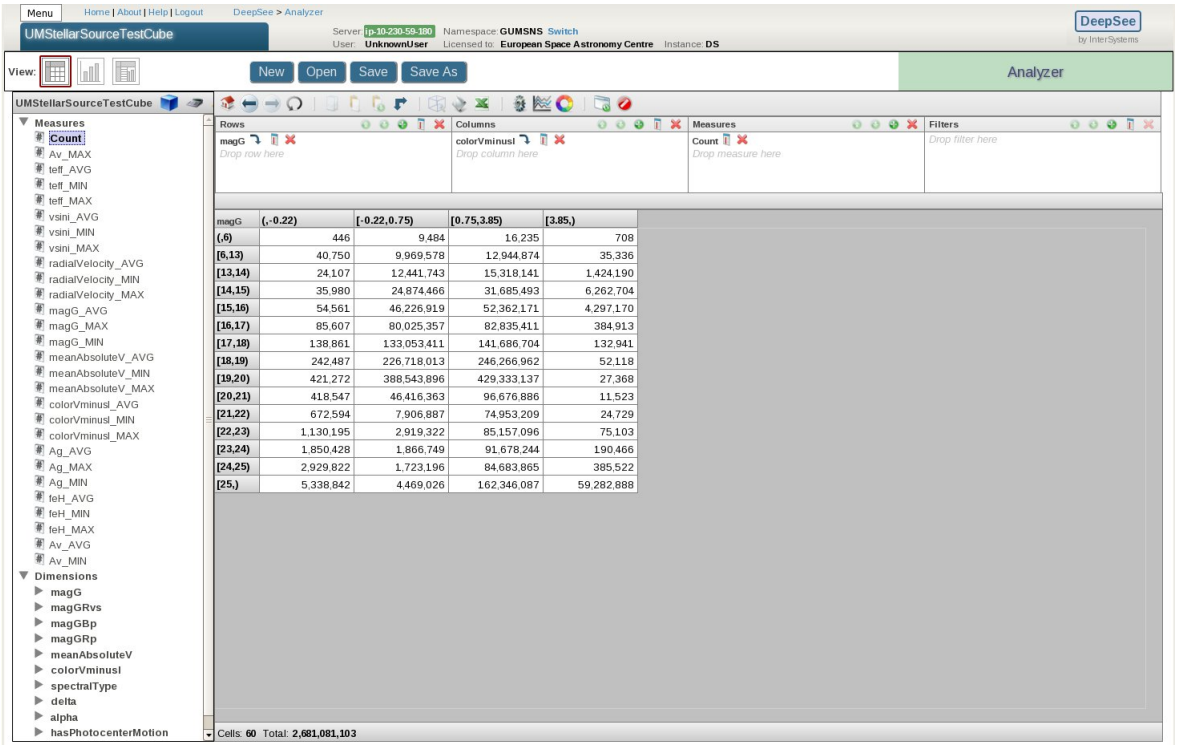


FIGURE 13: DeepSee Analyzer main view with UMStellarSourceTestCube.

as big as GUMS, DeepSee Analyzer will create the pivot tables in real time whenever the number of cells in the resulting pivot table is kept relatively small (up to 100 or so). For further information about the usage of DeepSee Analyzer, please refer to Intersystems documentation at `http://docs.intersystems.com/deepsee.html`.

Write further notes (possibly with some screenshots) about how to build a web portal from a pivot table and so on.

4.3 Test Benchmarks

Results of tests carried out within Intersystems DeepSee for GUMS data. Show as many use cases covered by DeepSee as possible (the ones also covered in GAT, etc).

4.4 Conclusions

Conclusions about Intersystems DeepSee with GUMS data volume.

DeepSee works fast with GUMS dataset whenever the amount of output cells is kept reasonably small.

DeepSee cannot work from an AS, so it may overload the DS when generating the cube or when building complex pivot tables for users. Furthermore, DeepSee does not give the expected performance when accessed in parallel by several users (even though the queries made are not very complex).

The string holding the range definition is limited in size. Therefore, the number of ranges to be defined in a field is limited, causing that ranges for fields like *alpha* or *delta* must be left at a coarse granularity (which is not the desirable thing to do).

DeepSee can generate histograms on the fly by using the different dimensions and ranges configured. However, if new categories or other ranges for existing categories are to be used, the hypercube must be recreated, and this task may take a couple of days or so to complete for the GUMS dataset volume in the Amazon EC2 instance described above. The conclusion of this is that Intersystems DeepSee is most valuable when used for categories whose values are discrete as they will not be changed over time, as opposed to the range definitions which are likely to change more than once depending on the data mining scenario.

There might be other scenarios where complex analysis has to be made over the fields of the data object so as to compute the measure to show in the pivot tables cells. A more thorough analysis of the capabilities offered by Intersystems DeepSee will have to be made to better understand the kind of calculations available for doing so.

A EBS single and RAID volumes I/O benchmark on Amazon EC2

Tab. 2 shows the performance obtained by Amazon EC2 EBS disks for the different configurations shown, where:

- *single*. Single EBS volume.
- *raid0_2disks*. RAID0 on two volumes.
- *raid0_4disks*. RAID0 on four volumes.
- *raid5*. RAID5 on three volumes.
- *raid10*. RAID10 on four volumes.
- *raid10_f2*. Special RAID10 f2 configuration.

The keywords used for describing the type of access are:

- *rndrd* for random read.
- *rndrw* for random read-write.
- *rndwr* for random write.
- *seqrd* for sequential read.
- *seqwr* for sequential write.

Response time in *Min*, *Max*, *Avg* and *95%* is expressed in *ms*. *95%* refers to the response time (in emphms as well) for 95% of total cases.

The filesystem used for these tests is *xfs* although similar conclusions are expected for *ext3* filesystem.

Configuration	Type	Threads	Rate	requests/s	Min	Max	Avg	95%
single	rndrd	1	1.496Mb	95.74	0.49	707.58	10.44	24.32
single	rndrd	4	1.7144Mb	109.72	0.74	279.8	36.44	66.28
single	rndrd	8	1.1727Mb	75.05	6.51	820.5	106.32	286.74
single	rndrd	16	1.0066Mb	64.42	39.14	961.32	248.12	504.14

Configuration	Type	Threads	Rate	requests/s	Min	Max	Avg	95%
single	rndrw	1	2.1997Mb	140.78	0.5	1352.35	7.1	21
single	rndrw	4	2.1138Mb	135.28	0.52	720.73	29.56	71.05
single	rndrw	8	1.9933Mb	127.57	1.02	688.37	62.69	186.45
single	rndrw	16	1.6991Mb	108.74	0.96	1278.89	146.49	414.87
single	rndwr	1	9.2364Mb	591.13	0.87	840.03	1.69	1.25
single	rndwr	4	4.7256Mb	302.44	0.92	266.45	13.22	23.98
single	rndwr	8	2.0775Mb	132.96	1.84	277.6	60.14	71.93
single	rndwr	16	1.7128Mb	109.62	3.93	242.59	145.78	160
single	seqrd	1	22.375Mb		0.49	1336.88	0.7	0.73
single	seqrd	4	40.438Mb	2588	0.5	214.7	1.54	2.25
single	seqrd	8	51.347Mb	3286.21	0.5	275.89	2.43	3.24
single	seqrd	16	39.258Mb	2512.48	0.51	460.88	6.37	28.42
single	seqwr	1	11.216Mb	717.85	0.91	1054.23	1.36	1.99
single	seqwr	4	11.773Mb	753.49	0.99	679.2	5.31	5.62
single	seqwr	8	12.514Mb	800.89	0.95	280.9	9.99	17.06
single	seqwr	16	13.449Mb	860.74	1.02	490.98	18.58	26.14
raid0_2disk	rndrd	1	2.5067Mb	160.43	0.49	1328.89	6.23	13.84
raid0_2disk	rndrd	4	4.2017Mb	268.91	0.49	687.31	14.87	32.71
raid0_2disk	rndrd	8	4.7691Mb	305.22	0.49	612.04	26.2	55.42
raid0_2disk	rndrd	16	4.9951Mb	319.69	0.49	413.44	50.03	107.1
raid0_2disk	rndrw	1	3.5892Mb	229.71	0.49	1272.62	4.35	12.34
raid0_2disk	rndrw	4	5.6414Mb	361.05	0.49	364.5	11.08	29.81
raid0_2disk	rndrw	8	6.1206Mb	391.72	0.5	372.72	20.42	51.45
raid0_2disk	rndrw	16	6.4457Mb	412.52	0.5	787.49	38.77	95.13
raid0_2disk	rndwr	1	12.871Mb	823.74	0.87	1314.99	1.21	1.12
raid0_2disk	rndwr	4	31.708Mb	2029.34	0.88	263.67	1.97	2.74
raid0_2disk	rndwr	8	41.618Mb	2663.58	0.88	379.95	3	4.5
raid0_2disk	rndwr	16	43.852Mb	2806.52	0.89	221.31	5.7	9.63
raid0_2disk	seqrd	1	20.661Mb	1322.28	0.48	1348.6	0.75	0.73
raid0_2disk	seqrd	4	39.132Mb	2504.44	0.49	225.59	1.6	2.36
raid0_2disk	seqrd	8	56.377Mb	3608.14	0.48	414.78	2.22	3.25
raid0_2disk	seqrd	16	65.213Mb	4173.65	0.49	254.61	3.83	5.2
raid0_2disk	seqwr	1	11.723Mb	750.26	0.87	481.28	1.3	1.37
raid0_2disk	seqwr	4	12.642Mb	809.1	1.08	476.74	4.94	5.24
raid0_2disk	seqwr	8	12.594Mb	806.02	1.23	544.31	9.92	10.11
raid0_2disk	seqwr	16	12.57Mb	804.45	0.92	552.43	19.88	29.35
raid0_4disk	rndrd	1	4.9586Mb	317.35	0.43	1150.42	3.15	11.86
raid0_4disk	rndrd	4	9.2366Mb	591.14	0.41	505.21	6.76	22.38
raid0_4disk	rndrd	8	11.183Mb	715.73	0.42	366.61	11.17	34.79
raid0_4disk	rndrd	16	10.437Mb	667.96	0.46	1558.01	23.94	77.52

Configuration	Type	Threads	Rate	requests/s	Min	Max	Avg	95%
raid0_4disk	rndrw	1	4.5071Mb	288.45	0.42	1361.59	3.46	13.13
raid0_4disk	rndrw	4	7.8167Mb	500.27	0.42	471.93	7.99	27.29
raid0_4disk	rndrw	8	9.4152Mb	602.57	0.42	313.77	13.27	47.29
raid0_4disk	rndrw	16	9.9225Mb	635.04	0.42	739.14	25.19	97.11
raid0_4disk	rndwr	1	12.811Mb	819.93	0.83	1151.15	1.22	1.49
raid0_4disk	rndwr	4	32.976Mb	2110.49	0.83	423.2	1.89	2.84
raid0_4disk	rndwr	8	49.149Mb	3145.53	0.85	269.4	2.54	5.49
raid0_4disk	rndwr	16	56.885Mb	3640.64	0.86	512.59	4.39	13.24
raid0_4disk	seqrd	1	22.398Mb	1433.49	0.41	1340.88	0.7	0.74
raid0_4disk	seqrd	4	38.967Mb	2493.9	0.42	143.95	1.6	2.22
raid0_4disk	seqrd	8	56.868Mb	3639.53	0.47	228.98	2.2	3.11
raid0_4disk	seqrd	16	45.352Mb	2902.54	0.46	255.51	5.51	5.36
raid0_4disk	seqwr	1	12.911Mb	826.29	0.85	263.93	1.18	1.37
raid0_4disk	seqwr	4	13.451Mb	860.84	0.99	269.41	4.64	5.37
raid0_4disk	seqwr	8	13.152Mb	841.76	1.04	293.2	9.5	10.85
raid0_4disk	seqwr	16	13.898Mb	889.49	0.99	284.39	17.98	25.59
raid5	rndrd	1	2.837Mb	181.57	0.47	992.51	5.51	15.11
raid5	rndrd	4	4.7228Mb	302.26	0.46	1026.42	13.23	36.69
raid5	rndrd	8	5.6219Mb	359.8	0.43	546.77	22.22	61.97
raid5	rndrd	16	6.1322Mb	392.46	0.47	962.75	40.74	111.42
raid5	rndrw	1	2.789Mb	178.5	0.43	1323.07	5.6	14.86
raid5	rndrw	4	4.3704Mb	279.7	0.45	380.68	14.3	36.32
raid5	rndrw	8	5.088Mb	325.63	0.46	552.7	24.55	60.75
raid5	rndrw	16	5.4255Mb	347.23	0.46	566.78	46.06	119.79
raid5	rndwr	1	2.2181Mb	141.96	1.35	1314.19	7.04	18.6
raid5	rndwr	4	2.315Mb	148.16	6.48	356.1	26.95	54.19
raid5	rndwr	8	2.2401Mb	143.36	2.94	870.65	55.79	215.64
raid5	rndwr	16	2.1925Mb	140.32	1.73	610.62	113.94	217.39
raid5	seqrd	1	17.675Mb	1131.22	0.42	1316.62	0.88	1.37
raid5	seqrd	4	34.255Mb	2192.35	0.46	217.92	1.82	3
raid5	seqrd	8	51.738Mb	3311.21	0.45	232.32	2.41	3.88
raid5	seqrd	16	39.916Mb	2554.61	0.45	256.57	6.26	5.7
raid5	seqwr	1	7.5288Mb	481.85	1.01	266.72	2.03	2.62
raid5	seqwr	4	7.8212Mb	500.56	1.12	632.63	7.99	24.24
raid5	seqwr	8	8.3782Mb	536.2	1.74	422.65	14.91	48.94
raid5	seqwr	16	8.6235Mb	551.9	1.72	308.87	28.98	44.33
raid10	rndrd	1	2.5282Mb	161.8	0.48	1289.18	6.18	17.06
raid10	rndrd	4	8.0656Mb	516.2	0.42	417.43	7.75	22.31
raid10	rndrd	8	9.0941Mb	582.02	0.46	395.92	13.74	41.95
raid10	rndrd	16	9.4685Mb	605.98	0.46	834.45	26.32	84.04

Configuration	Type	Threads	Rate	requests/s	Min	Max	Avg	95%
raid10	rndrw	1	3.498Mb	223.87	0.44	1363.39	4.47	14.53
raid10	rndrw	4	5.8657Mb	375.4	0.43	371.45	10.65	28.59
raid10	rndrw	8	6.3957Mb	409.33	0.48	377.68	19.54	51.42
raid10	rndrw	16	6.2767Mb	401.71	0.46	618.12	39.81	113.78
raid10	rndwr	1	9.4426Mb	604.32	0.98	1328.79	1.65	2.87
raid10	rndwr	4	22.996Mb	1471.77	1.02	304.27	2.72	4.42
raid10	rndwr	8	27.602Mb	1766.52	1.06	269.39	4.53	8.86
raid10	rndwr	16	30.235Mb	1935.03	1.02	463.2	8.27	18.1
raid10	seqrd	1	21.74Mb	1391.34	0.41	1267.69	0.72	0.74
raid10	seqrd	4	39.049Mb	2499.14	0.45	130.36	1.6	2.26
raid10	seqrd	8	55.907Mb	3578.06	0.46	214.89	2.23	3.3
raid10	seqrd	16	49.603Mb	3174.56	0.45	263.21	5.04	12.6
raid10	seqwr	1	10.936Mb	699.9	1	266.42	1.39	1.75
raid10	seqwr	4	10.828Mb	693	1.14	273.01	5.77	6.62
raid10	seqwr	8	10.808Mb	691.74	1.16	294.39	11.56	15.98
raid10	seqwr	16	11.098Mb	710.28	1.18	288.38	22.52	37.18
raid10_f2	rndrd	1	2.2497Mb	143.98	0.47	1554.59	6.94	19.26
raid10_f2	rndrd	4	4.9531Mb	317	0.45	360.1	12.62	35.3
raid10_f2	rndrd	8	4.4123Mb	282.39	0.47	414.79	28.32	80.21
raid10_f2	rndrd	16	4.3907Mb	281	0.48	990.51	56.91	153.2
raid10_f2	rndrw	1	2.6882Mb	172.04	0.44	1281.14	5.81	18.76
raid10_f2	rndrw	4	3.63Mb	232.32	0.42	545.59	17.21	57.83
raid10_f2	rndrw	8	4.0247Mb	257.58	0.47	590.84	31.06	107
raid10_f2	rndrw	16	5.0853Mb	325.46	0.45	860.57	49.12	145.95
raid10_f2	rndwr	1	10.83Mb	693.14	0.99	1554.88	1.44	1.74
raid10_f2	rndwr	4	19.828Mb	1269.02	1.01	347.93	3.15	5.16
raid10_f2	rndwr	8	24.084Mb	1541.36	1	347.19	5.19	10.13
raid10_f2	rndwr	16	25.164Mb	1610.47	0.98	443.36	9.93	20.71
raid10_f2	seqrd	1	19.227Mb	1230.52	0.41	1435.02	0.81	0.74
raid10_f2	seqrd	4	31.433Mb	2011.68	0.46	291.9	1.99	3.03
raid10_f2	seqrd	8	46.719Mb	2990	0.45	638.28	2.67	4.15
raid10_f2	seqrd	16	43.167Mb	2762.7	0.47	471.4	5.79	14.29
raid10_f2	seqwr	1	9.5123Mb	608.78	1	265.07	1.6	1.99
raid10_f2	seqwr	4	9.87Mb	631.68	2.38	968.84	6.33	6.99
raid10_f2	seqwr	8	9.5382Mb	610.45	1.17	1523.26	13.1	18.1
raid10_f2	seqwr	16	9.733Mb	622.91	1.19	1763.71	25.68	39.08

Table 2: EBS I/O benchmark for different disk configurations on Amazon EC2