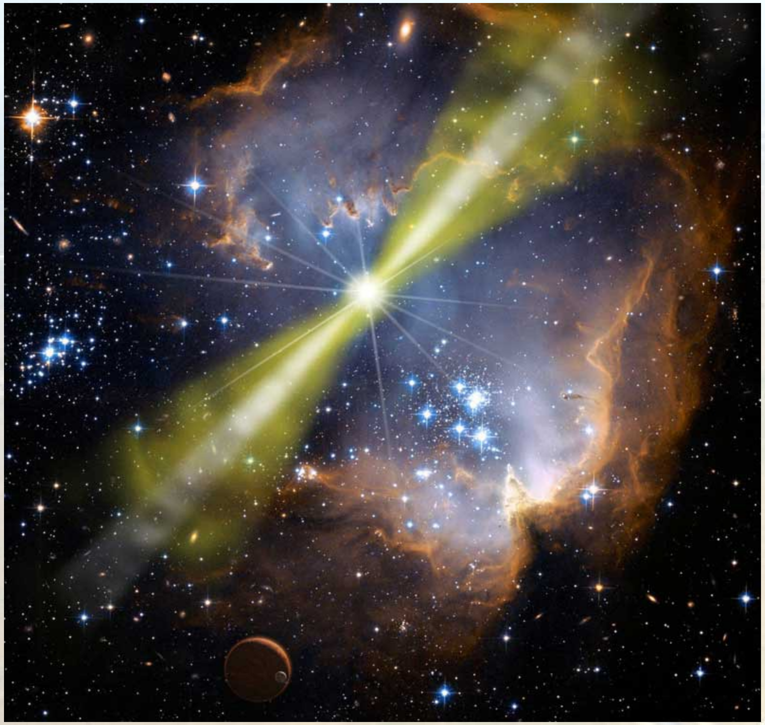
Status of and challanges in Virgo computing

Gergely Debreczeni

Wigner RCP
Virgo Computing Coordinator

(Debreczeni.Gergely@wigner.mta.hu)

Content

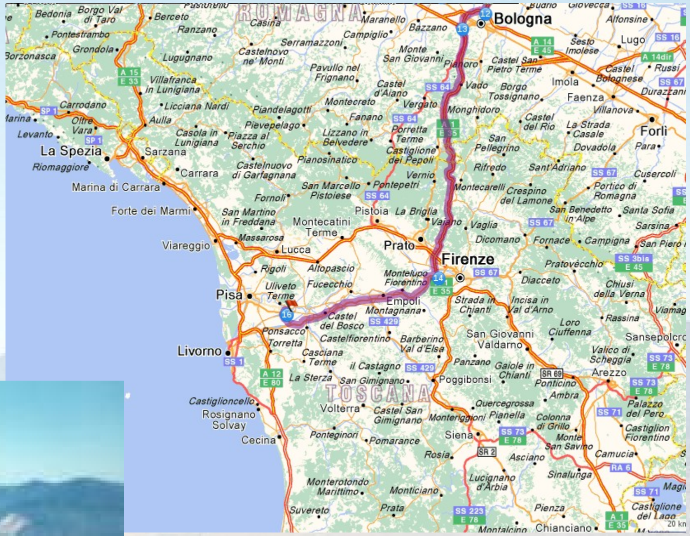


Artists's view of a gramma-ray burst

Credit: NASA/Swift/Mary Pat Hrybyk-Keith and John Jones

- **Virgo experiment**
    - A one slide introduction to the Virgo experiment
- **Computing model walkthrough**
    - Scale, type and strategies in Virgo computing
- **Data transfer**
    - Transfer of measurement data to CCs
- **Analysis types**
    - Computing needs of anaalysis workflows
- **Pipeline execution problems**
    - Problems related to pipeline execution
- **GPU perspectives**
    - How GPUs can help us in cheaper computing
- **Cloud solution**
    - Why a computing cloud would be excellent for Virgo
- **Future plans**
    - Things we would like to work on in the future

The Virgo experiment



• The Virgo detector is located in the site of the European Gravitational Observatory (EGO) in Cascina, near Pisa, Italy.

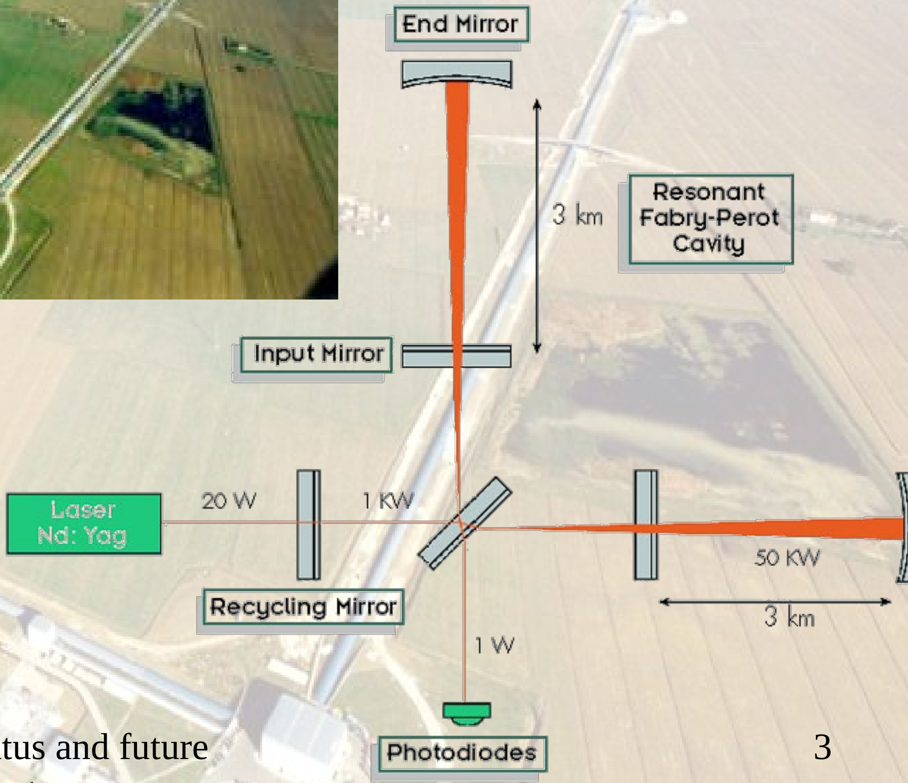
• Construction finished in 2003

• It is now a european collaboration including France, Italy, Hungary, Netherland, Poland

• Working together with LIGO (Laser Interferometer Gravitational-wave Observatory), synchronized observations and coordinated analysis

• So far, approixmately c.c 20 month of data taking

• Currently under upgrade, will start to collect scientific data in early 2016



Computing model in 1 slide

- Data is taken by the 3 detector
- Online (low-latency) analysis happens on the measurement site.
- Data is stored at site temporarly in a circular buffer (typicaly for 6 month)
- All data is transferred with c.c. 1 day latency to external CCs and stored in 3 different location (Lyon, CNAF, Ligo site)
- Virgo uses its two main CCs and the INFN grid sites, while LIGO uses dedicated Condor clusters and XEDE supercomputer resources
- Offline analysis based on Condor DAG, Pegasus DAX and shared file system based workflows and/or simple EMI Grid submission mechnisms. No unified job scheduler is used over the collaboration.
- Analysis code must undergo serious review, coordinated prioritisation and optimisation efforts. Reviewed code is tagged, freezed.

Type of measurement data

- The measurement data is a time series, sampled at 20 kHz, then downsampled to 16 kHz and 4 kHz.
- There are hundreds of auxiliary enviromemntal channels, some of them with much lower frequencies
- Amount of data is a few hundred TB / yr but its arithmetic complexity is much higher than that of the HEP experiments.
- Depending on the source to be examined / discovered many different kind of analysis is crunching this data with computing requirements differing by order of magnitued.
- Different analysis requiring different input data size and computing architectures
- Gravitational wave analysis is **compute intensive** not data intensive

Data transfer

- Measurement data produced in Cascina on the Virgo site
- **Low latency** online analysis happens in place and data is stored temporarly on site using a circular buffer of length of several month.
- For **offline analysis** this data must be transferred to the computer centers.
- For this purpose a transfer tool was developed by EGO.
- It uses the
  - lcg-tools to transfer data to CNAF
  - and the irod client installation for IN2P3
- The amount of data is not overwhelming, in principle data transfer should not be a challange
- Data from LIGo detectors are also get copied to our CCs. This data consist(ed) of small file which must be merged to bigger ones in order to fit better for HPSS storage. Now this problem had been solved

Data transfer issues

- Virgo is using different data transfer tools for different data transfers. Each of them is easy and works reliably, however the need for using multiple backend needs extra manpower and development.
- A preffered solution would be to use the LDR (LIGO Data Replicator) data transfer framework all over in the LIGO-Virgo collaboration which cann communicate with legacy GSIFTP backends and perform reliable data transfer.
- GSIFTP is not available in IN2P3 which is a problem, probably will be solved soon

Data and file and metadata catalogs

- Some analysis is using the LFC (Lightweight File Catalog). Its use is easy and we found no problem with it.
- The DIsckcache software is used to catalog files available on a given site and respond to queries of the pipelines with physical location of files.
- Typical query includes GPS times, detector name, channel name.
- No file metadata catalog is used so far, there is not too much need of that. Many information is in file names and can be easily queried.

Analysis pipelines

- **CW** (Continuos waves) - Rotating, assymetric neutron stars
  - The most compute intense pipeline, practically can consume all available resources. Sensitivity goes like 1/sqrt(T), where T is the duration of data chunk in question. As a result one must restrict parameter space explored -> scientific limitation.
- **CBC** (Compact Binary Coalescence) - Gravitational waves emitted by coalescing binarie neutron stars or binary black holes
  - Very compute intensive, theoretical templates are tested againts the measurement data by means of matched filter. With the decrease of low frequency cutoff compute costs grows exponentially.
- **Burst** - Explosions, supernovas, unmodelled tranzient sources
  - Very similar but more generic than CBC. Sensitivity is comparable (~c.c 30% less).
- **Stochastic** - Search for stochastic gravitational waves of galactic or primordial origin
  - Important from physics point of view, but has negligible compute cost.

Analysis pipeline problems

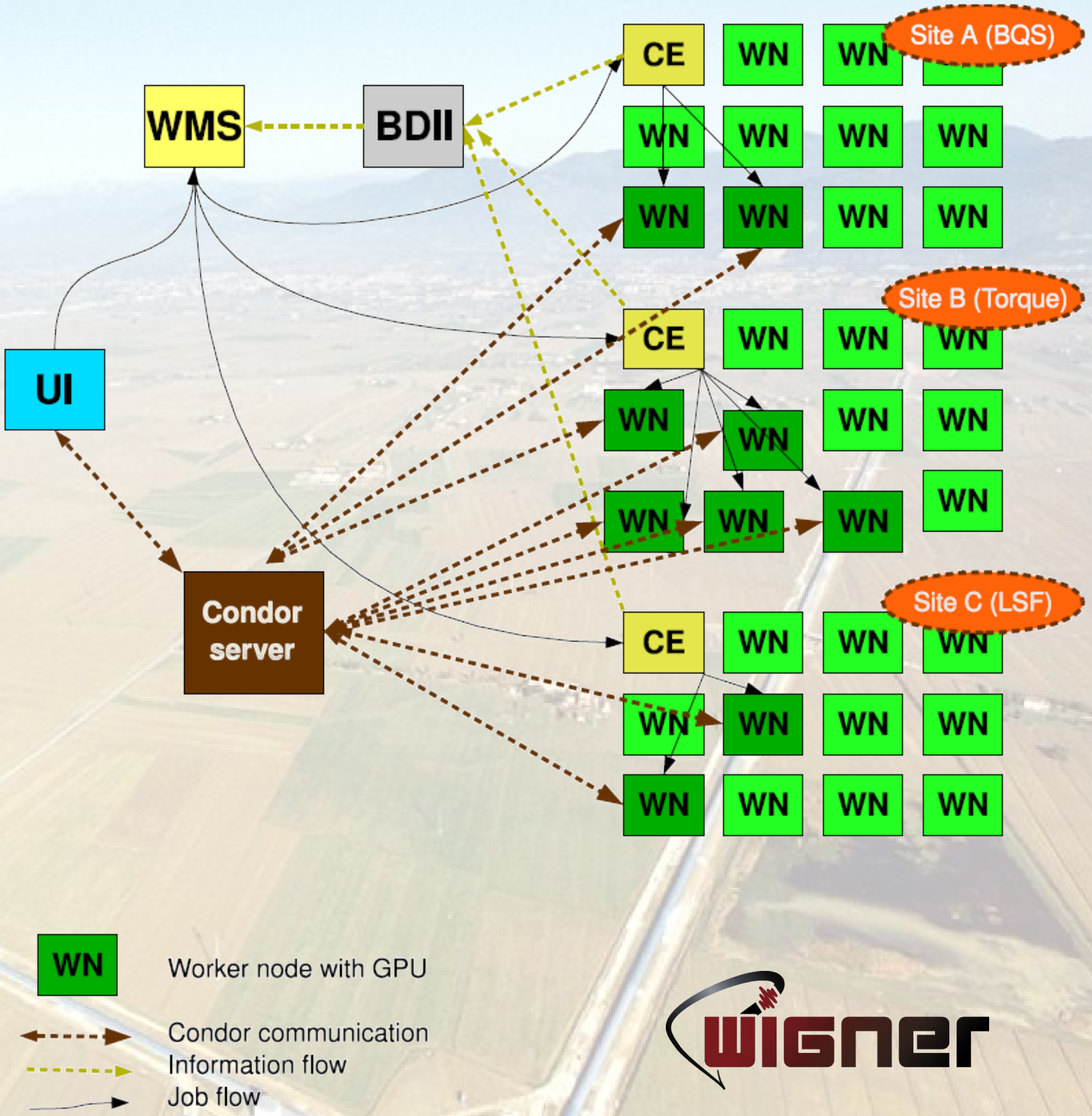
- LIGO and Virgo collaboration is working closely together.
- However LIGO collaboration is larger by a factor of > x10
- As a result many important pipeline development is dominated by LIGO colleagues and are tailored to LIGO resources
- As a consequance those pipelines cannot be executed out of the box on pur resources, but requires quite some porting effort.
- This effort is not a one time action, but needs continouse attention -> very expensive in terms of time and manpower...Virgo cannot afford.
- Many attempts have been made to overcome this difficulty including
  - the set up of a pilot pool framework
  - using the Pegasus scheduler
  - examining the possibility of using the Dirac jobmission framework
  - thinking on virtualized Condor cluster, i.e. a Cloud

The Virgo Pilot Pool - I

The Virgo Pilot Pool properties:

- Homogen infrastructure over the inhomogen Grid
- Less administrative interaction/delay
- User transparent mechanisms
- Low latency submission
- Global priorities

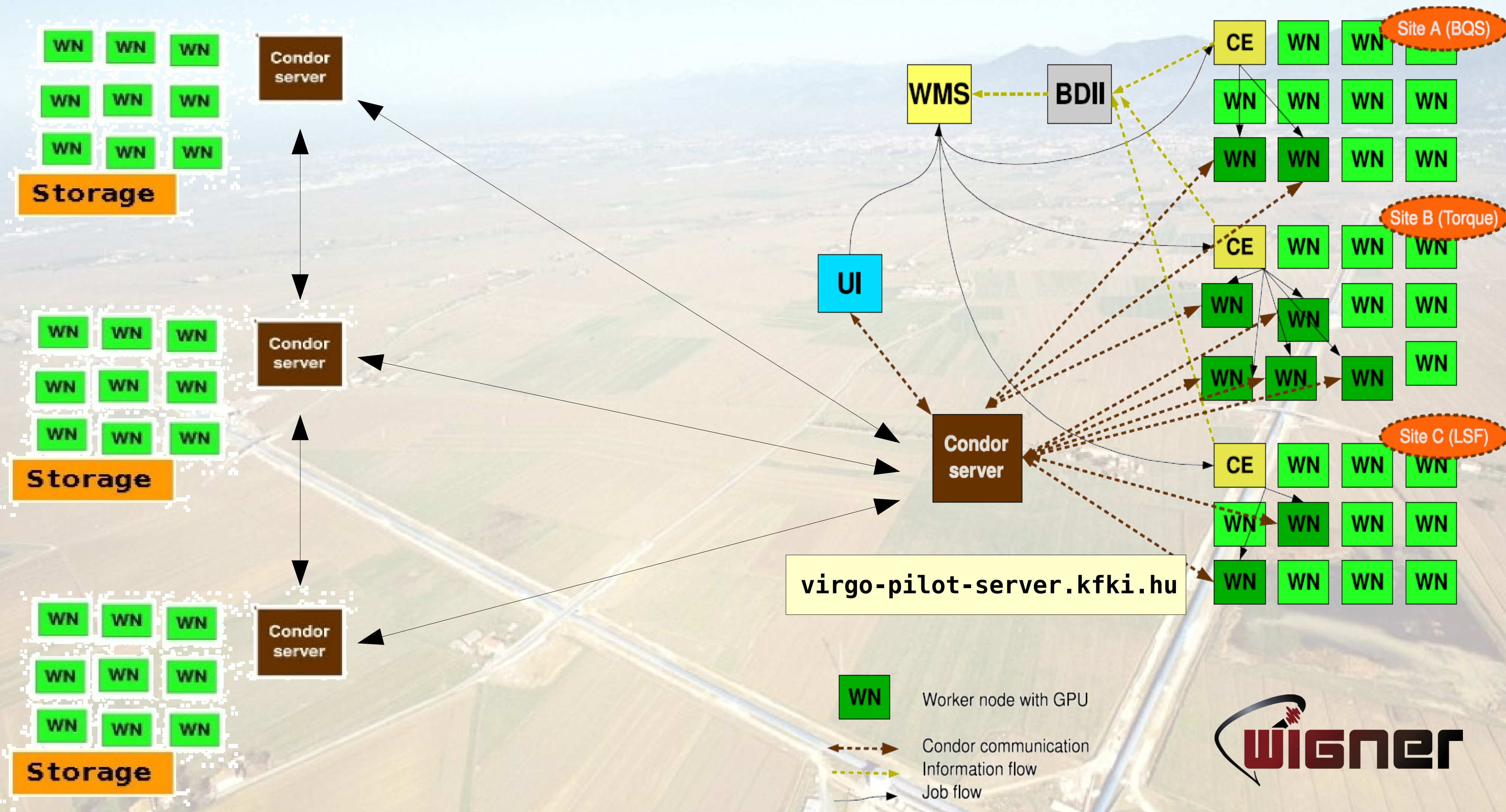
- Late-binding to resources
- No stucked-in jobs
- Improved job failure rate due to pilot prechecks
- Interactive login
- Smooth interaction interoperability with LDG/OSG.



The Virgo Pilot Pool - II

Ligo clusters

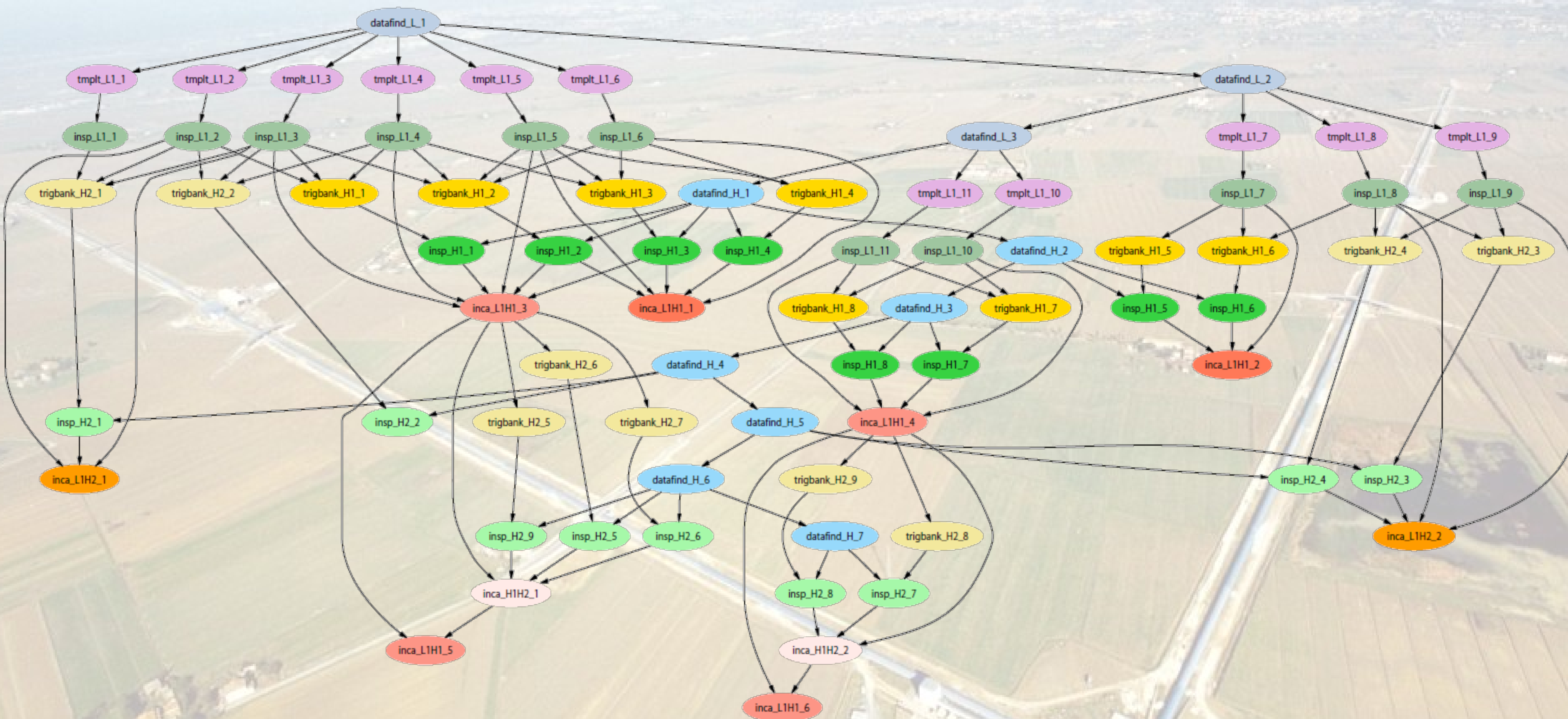
Virgo EGI sites



The Virgo Pilot Pool - III

- Mapping of abstract workflows like DAGs/DAXes to the Grid is now easily possible with the Virgo Pilot server.

- Complex and relational workflow handling is/was missing from Cream/WMS.



GPUs for analysis

- Many search algorithm can be accelerated by making use of operation level paralellizability offered by various many-core hardwares such as GPUs. Such examples are:
  - FFT, vector operations, reduce, max finding, clustering **in CBC analysis pipelines**
  - FFT, 2D thresholding, Differential Hough map creation, integration, peak finding **in CW analysis**
- There are multiple tool developed to allow easier use of GPUs by less advanced programmers, such as:
  - GWTools - An OpenCL based templates C++ generic algorithm library for GW searches
  - pyCBC - CUDA based set of Python algorithm used in CBC analysis
  - CB - Compute Backend - offers a unified host code for CUDA and OpenCL, so there is no need to write the code twice for NVidia and AMD cards
- GPUs will play crucial role in the following years probably even for the discovery
- Typical full-pipeline accelerations experienced are ranging from x30 to x120

Cloud 4 Virgo

- Out of the above possibilities each of them has some serious drawbacks except the Cloud solution
- Having an OpenStack based virutal Cloud Condor installation at CNAF would solve almost all our problem including,
  - pipeline porting
  - architectural difference
  - training of people and the must to learn multiple submission mechanisms
  - real sharing of our resources
  - easy of GPU access and GPU - CPU matching, allocation problems
  - better monitoring of user jobs

Optimisation, prioritisation

- The LIGO - Virgo analysis software stack went through on a serious process of review, benchmarking and optimisation.
- The process was triggered by the NSF review of LIGO request for XEDE resources
- It has a very positive effect on the quality, organisation and performance of the code used for analysis in the Collaborations.
- Analysis type based compute resource request estimation, logging and accounting and prioritisation is just under introduction in the collaborations
- A common unit of measure for called „Service Unit” SU has been introduced, since we observed that HS06 numbers are not alway accurate enough in reflecting the ratios of performances of a specific CPU cores for specific analysis. 1 SU corresponds to the performance of an Intel Xeon E5-2650 core.
- LVCN - the LIGO - Virgo Computing Network was estableshed

LVCN supply table

LIGO-Virgo O1 CPU/GPU/MIC Inventory (March 2015)

DEPLOYED SYSTEMS					INDIVIDUAL DEVICE STATISTICS												SYSTEM TOTAL				RELATIVE PERF					AVAILABLE CORES (E5-2670 equivalent)										
Tag	SITE	Org	Arch	Model	Qty	Peak	SPECfp	FFT	pycbc	pycbc	gstlal	Core	Clock	Cache	max flop	Mem	Peak	CPU	GPU	Mem	SPECfp	Full Device Performance				Dev	Dev	Prod	Prod	Peak	O1	O2	O3	Deployed	EOL	Comment
						GLOPS		GFLOPs	(4k)	(2k)	(2k)		MHz	MB	per cycle	GB	GLOPs	core	core	GB		FFT	pycbc(4k)	pycbc(2k)	gstlal(2k)	%	core	%	core	core	rank	rank	rank	date		
CIT	CIT	Lab	CPU	Opteron 275	662	18	15					2	2,200	2	4	2	11,651	1,324		1,324	0.06					100	331	0	0	0				02/2006	2015	
CIT			CPU	Opteron 2376	328	74	54					4	2,300	6	8	8	24,141	1,312		2,624	0.23					100	590	0	0	0	1			03/2009		
CIT			CPU	Xeon E5-2670	96	333	240	102.9	43,875	97,488	44,000	8	2,600	20	16	64	31,949	768		6,144	1.00	1.00	1.00	1.00	1.00	0	0	100	768	768	1	1	1	12/2012	8/2013	
CIT			CPU	Xeon E3-1241v3	200	448	153		27,903	71,052		4	3,500	8	32	32	89,600	800		6,400	0.64		0.64	0.73		0	0	100	1,020	1,020	1	1	1	4/1/2015		aLIGO FY15 procureme
CIT			GPU	Tesla K10	64	4,580		378				3,072	745			8	293,120		196,608	512		3.67				0		100						12/2012		
CIT			MIC	Xeon Phi 5110P	32	2,022						60	1,053			8	33,696	1,920		256						0		100						8/2013		
OBS	LLO		CPU	Opteron 2376	504	74	54					4	2,300	6	8	8	37,094	2,016		4,032	0.23					100	907	0	0	0	1			03/2009		
OBS			CPU	Xeon E3-1241v3	100	448	153					4	3,500	8	32	32	44,800	400		3,200	0.64					0	0	100	510	510	1	1	1	4/1/2015		aLIGO FY15 procureme
OBS	LHO		CPU	Opteron 2376	504	74	54					4	2,300	6	8	8	37,094	2,016		4,032	0.23					100	907	0	0	0	1			03/2009		
OBS			CPU	Xeon E3-1241v3	100	448	153					4	3,500	8	32	32	44,800	400		3,200	0.64					0	0	100	510	510	1	1	1	4/1/2015		aLIGO FY15 procureme
HAN	HAN	LSC	CPU	Xeon X3220	1,176	38	40					4	2,400	8	4	8	45,158	4,704		9,408	0.17					50	784	0	0	0	1			03/2008	2014/5	
HAN			CPU	Xeon E3-1220v3	1,820	397	137	40	28,511	64,397		4	3,100	8	32	16	722,176	7,280		29,120	0.57	0.39	0.65	0.66		0	0	50	4,156	8,311	1	1	1	02/2014		
HAN			CPU	Xeon E3-1231v3	270	435	151	55			37,600	4	3,400	8	32	16	117,504	1,080		4,320	0.63	0.54			0.85	0	0	50	680	1,359	1	1	1	02/2014	0	
HAN			CPU	Xeon E5-1650v2	52	336	220					6	3,500	12	16	64	17,472	312		3,328	0.92					0	0	50	191	381	1	1	1	02/2014		
HAN			CPU	Xeon E5504	62	64	57					4	2,000	4	8	12	3,968	248		744	0.24					0	0	50	59	118				01/2010	2015?	
HAN			GPU	Tesla C1060	116	933						240	1,300			4	108,228		27,840	464														01/2010	2015?	Atlas GPU numbers cor
HAN			GPU	Tesla C2050	101	1,030						448	1,150			3	104,030		45,248	303														01/2010	2015?	
HAN			GPU	GT640	459	691						384	900			2	317,169		176,256	918														03/2014		
HAN			GPU	GTX750	270	1,044						512	1,080			2	281,880		138,240	540														06/2014		
SYR	SYR		CPU	Xeon X3220	80	38	40					4	2,400	8	4	8	3,072	320		640	0.17					50	53	0	0	0	1			10/2007	10/2015	Sugar: Confirm availabi
SYR			CPU	Xeon X5650	667	128	125		23,808	43,864		6	2,668	12	8	24	85,419	4,002		16,008	0.52		0.54	0.45		0	0	100	2,779	2,779	1	1	1	2015	2,018	Crush: MoU not signed
SYR			GPU	GTX 580	172	1,581						512	722			2	271,949		88,064	258						50		0			1			10/2010	10/2015	
UWM	UWM		CPU	Xeon X5650	320	128	125					6	2,670	12	8	24	41,011	1,920		7,680	0.52					100	1,333	0	0	0	1			1/2010	2015	
UWM			CPU	Xeon E5-2650	184	256	210					8	2,000	20	16	32	47,104	1,472		5,888	0.88					0	0	90	1,159	1,288	1	1	1	9/1/2012	9/2017	
UWM			GPU	GTX 690	4	5,620						3,072	1,000			2	22,480		12,288	8						0		100						9/2012	9/2017	
UWM			GPU	GTX 580	4	1,581						512	722			2	6,324		2,048	6						0		100						9/2012	9/2017	
UWM			GPU	Tesla K10	10	4,580						3,072	745			8	45,800		30,720	80						0		100						9/2012		
UWM			GPU	Tesla M2075	10	1,030						448	1,150			6	10,300		4,480	60						0		100						9/2012	9/2017	
IUCAA	IUCAA		CPU	Xeon E5-2670	60	333	240					8	2,600	20	16	64	19,968			3,840	1.00					25	120	0	0	0	2	2	2	5/2013	8/2017	
Cardiff	Cardiff		CPU	Xeon X5660	128	134	125					6	2,800	12	8	48	17,203	768		6,144	0.52					0	0	67	357	533	1	1		12/2012	12/2016	
Cardiff			CPU	Xeon E5-2690v3	100	998	400					12	2,600	30	32	64	99,840	1,200		6,400	1.67					0	0	67	893	1,333	1	1	1	01/2015		
KGWG	KGWG		CPU	Xeon E5645	30	115	106					6	2,400	12	8	12	3,456	180		360	0.44					50	53	0	0	0				01/2012	2016?	
KGWG			CPU	Xeon E5645	40	115	106					6	2,400	12	8	24	4,608	240		960	0.44					50	71	0	0	0				01/2012	2016?	
KGWG			CPU	Xeon X5650	4	128	125					6	2,670	6	8	24	513	24		96	0.52					50	8	0	0	0				01/2010	2016	
KGWG			CPU	Xeon X5650	18	128	125					6	2,670	6	8	12	2,307	108		216	0.52					50	38	0	0	0				01/2010	2016	
CNAF	CNAF	Virgo	CPU	Heterogeneous	125	166	120					8	2,600		8	64	20,800	1,000		8,000	0.50					0	0	80	400	500	1			01/2011	8/2013	Remaining 20% availbl
Lyon	Lyon		CPU	Xeon E5645	50	115	210					6	2,400	12	8	96	5,760	300		4,800	0.88					0	0	100	350	350	1			01/2013	2018	
Rome	Rome		CPU	Xeon E5410	52	154	70					8	2,400	12	8	8	7,987	416		416	0.29					0	0	80	97	121	1			01/2008		
Wigner	Wigner		CPU	Opteron 6376	12	147	200					16	2,300	16	4	32	1,766	192	192	384	0.83					0	0	90	72	80	1			1/2013		
Wigner			GPU	AMD 7970	16	4,301						2,048	1,000			3	68,816			48						0		90						1/2013		
Nikhef	Nikhef		CPU	Xeon E5520	45	72	150					4	2,260	8	8	24	3,254	180		1,080	0.63					0	0	100	225	225	1			?		
Polgraw	Polgraw		CPU	Xeon E5-2665	16	307	235					8	2,400	20	16	64	4,915	128		1,024	0.98					0	0	100	125	125	2			1/2015	2018	To be used for Virgo CW
Polgraw			CPU	Opteron 6172	20	101	150					12	2,100	12	4	32	2,016	240		640	0.63					0	0	100	100	100	2			1/2012	2016	To be used for Virgo CW
Polgraw			CPU	Xeon E5645	80	115	106					6	2,400	12	8	24	9,216	480	480	1,920	0.44					0	0	100	283	283	2			?	2016	To be used for Virgo CW
Polgraw			GPU	GTX Titan	4	4,709						2,688	1,000			24	18,836		10,752	96						0		100						1/1/2014	?	To be used for Virgo CW
EGO	EGO		CPU	Xeon E5-2650	13	256	210					8	2,000	20	16	128	3,328	104		1,664	0.88					0	0	100	91	91	1			1/1/2015	9/2017	
Tag	SITE	Org	Arch	Model	Qty	Peak	SPECfp	FFT	pycbc	pycbc	gstlal	Core	Clock	Cache	flop/core	Mem	Peak	CPU	GPU	Mem	SPECfp	Full Device Performance				Dev	Dev	Prod	Prod	Peak	O1	O2	O3	Deployed	EOL	Comment
						GLOPS		GFLOPs	(4k)	(2k)	(2k)		MHz	MB	per cycle	GB	GLOPs	core	core	GB		FFT	pycbc (2k)	pycbc(4k)	gstlal(2k)	%	core	%	core	core	rank	rank	rank			

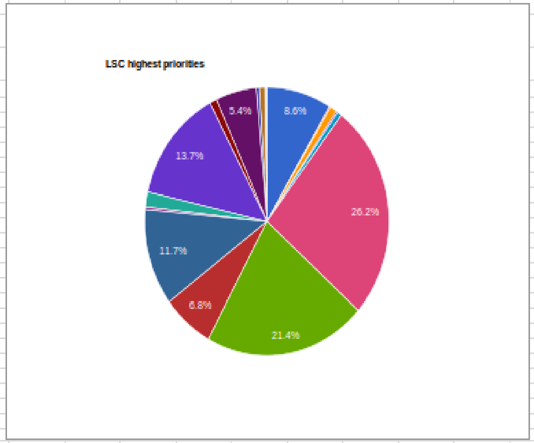
LVCN pipeline needs

fx Highest Priority

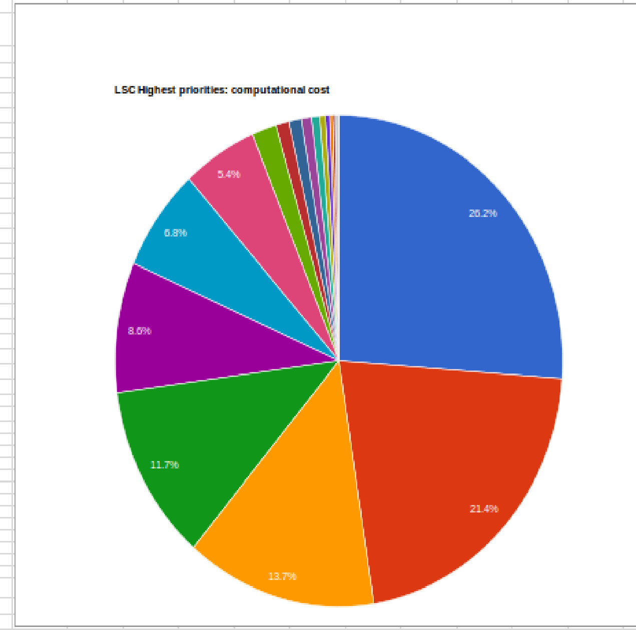
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Highest P		searches most likely to make detections or yield significant astrophysical results																						
2	RYBG					Unique ID	Storag	Development/testing (SU)			Simulations (SU)			Production (SU)			Additional for detection followup			duction timing rel. to run (mon		latency rel. to (	Contact name	MSU	
3	Code	Group	Search	Pie Name	Software	job schedu	Tb/yea	2015	2016	2017	2015	2016	2017	O1	O2	O3	O1	O2	O3	Start	End	taking (hours)	(albert.einstein)	2017	
4	B	BURST	All-sky burst (64-2000 Hz) - offline, 10	Burst All-sky	cWB2G	allsky.cwbof	200	115,000	110,000	110,000	23,000	38,000	76,000	117,000	486,000	730,000	1,170,000	4,860,000	7,300,000				sergey.klimenko	8.2	Burst All-sky
5	B	BURST	All-sky burst HF (2kHz-5kHz) 10000 la	Burst All-sky	CWB2G	allsky_hf.cw	5	135,000	160,000	160,000	28,000	51,000	102,000	145,000	689,000	1,050,000	1,450,000	6,890,000	10,500,000				sergey.klimenko	11.8	Burst All-sky
6	B	BURST	All-sky burst - online, 5000 lags	Burst low latency	CWB2G	allsky.cwbo	20	70,000	65,000	65,000	12,000	18,000	38,000	58,000	243,000	365,000	-	-	-				sergey.klimenko	0.5	Burst low latency
7	Y	BURST	All-sky validation pipeline, 300 lags, f	Burst All-sky	omicron+L	allsky.omicr	1	165,000	230,000	86,000	83,000	115,000	86,000	328,000	920,000	1,380,000	328,000	920,000	1,380,000				erik.katsavounidis	2.9	Burst All-sky
8	R	BURST	All-sky validation pipeline (EP)*	Burst All-sky	EP		0.5	23,000	9,000	3,000	6,000	5,000	3,000	24,000	37,000	55,000	24,000	37,000	55,000				chris.pankow	0.1	Burst All-sky
9	R	BURST	EM follow-up alerts	Burst All-sky	dedicated	???	0.2	-	-	-	-	-	-	-	-	-	-	-	-				peter.shawhan	-	Burst All-sky
10	B	BURST	Burst parameter estimation followups	Burst detection params	BayesWav	paramest.ba	5	160,000	160,000	160,000	240,000	360,000	360,000	123,000	334,000	500,000	300,000	900,000	1,400,000				tyson.littenberg	2.4	Burst detection params
11	Y	BURST	Burst parameter estimation followups	Burst detection params	LIB	paramest.lib	0.1	1,000	1,000	1,000	-	-	-	60	170	250	-	-	-				erik.katsavounidis	0.0	Burst detection params
12		**BURST**	**All-sky burst total**				**231.8**	**669,000**	**735,000**	**585,000**	**392,000**	**587,000**	**665,000**	**795,060**	**2,709,170**	**4,080,250**	**3,272,000**	**13,607,000**	**20,635,000**						
13	R	BURST	SNEWs triggered supernova search (	Burst astro events	raven	snews.raver	****	-	-	-	400	500	500	1,100	3,500	5,500	-	-	-				alex.urban	0.0	Burst astro events
14	Y	BURST	SNEWs triggered supernova search (	Burst astro events	CWB2G	snews.cwbo	0.1	16,000	2,000	2,000	100	100	100	100	100	100	-	-	-				michele.zanolin	0.0	Burst astro events
15	B	BURST	SNEWs triggered supernova search (	Burst astro events	X-Pipeline	snews.x	0.1	-	-	-	32,000	92,000	92,000	600	92,000	92,000	-	-	-				patrick.sutton	0.2	Burst astro events
16	R	BURST	SNEWs triggered supernova search (	Burst astro events	TBD	TBD	0.1	16,000	2,000	2,000	32,500	92,600	92,600	1,800	95,600	97,600	-	-	-				*See Burst tab*		Burst astro events
17	Y	BURST	Astrophysically significant NS transien	Burst astro events	X-Pipeline	ns.x	0.1	500	-	-	50	100	100	50	100	100	-	-	-				james.clark	0.0002	Burst astro events
18	Y	BURST	GRB triggered burst search - medium	Burst astro events	X-Pipeline	grb.xonline	0.4	-	-	-	2,000	6,000	6,000	6,000	32,000	48,000	-	-	-				patrick.sutton	0.1	Burst astro events
19	Y	BURST	HEN triggered search - online	Burst astro events	CWB2G	hen.cwbonli	0.1	-	-	-	400	500	500	1,100	3,500	5,500	-	-	-				imre.bartos	0.0	Burst astro events
20		**BURST**	**Outstanding astrophysical searches total**					**16,500**	**2,000**	**2,000**	**34,950**	**99,200**	**99,200**	**8,950**	**131,200**	**151,200**	**-**	**-**	**-**						
21	R	BURST	Cosmic string - offline	Cosmic strings	dedicated	cs.cs	0.3	-	-	-	1,000	1,700	2,600	4,000	11,000	17,000	128,000	356,000	534,000				florent.robinet	0.6	Cosmic strings
22			*Total Highest Priority*					*685,500*	*737,000*	*587,000*	*427,950*	*687,900*	*766,800*	*808,010*	*2,851,370*	*4,248,450*	*3,400,000*	*13,963,000*	*21,169,000*						
23	High Prior		promising extensions of the highest priority goals that explore larger regions of parameter space or can further the science potential of LIGO and Virgo																						
24	RYBG					Unique ID	Storag	Development/testing (SU)			Simulations (SU)			Production (SU)			Additional for detection followup			duction timing rel. to run (mon		latency rel. to (	Contact name		
25	Code	Group	Search	Pie Name	Software	job schedu	Tb/yea	2015	2016	2017	2015	2016	2017	O1	O2	O3	O1	O2	O3	Start	End	taking (hours)	(albert.einstein)		
26	B	BURST	GRB triggered search - offline **	Burst GRB	X-Pipeline	grb.xoffline	0.4	-	-	-	4,000	41,000	41,000	8,000	132,000	262,000	2,000	31,000	31,000				patrick.sutton	0.3	Burst GRB
27	Y	BURST	HEN triggered search - offline	High-energy neutrinos	CWB2G	hen.cwboffli	0.1	-	-	-	400	500	500	1,100	3,500	5,500	-	-	-				imre.bartos	0.0	High-energy neutrinos
28	B	BURST	IMBBH burst search - offline * 2500 la	Burst BBH	CWB2G	imbh.cwbof	****	115,000	110,000	110,000	23,000	38,000	76,000	39,000	190,000	308,000	390,000	1,900,000	3,080,000				sergey.klimenko	3.6	Burst BBH
29	Y	BURST	Optical supernova triggered - offline	Burst SuperNovae	CWB2G	sn.cwboffin	0.1	-	100,000	100,000	-	8,000	24,000	-	1,000	20,000	-	-	-				michele.zanolin	0.1	Burst SuperNovae
30	B	BURST	Optical supernova triggered - offline (V	Burst SuperNovae	X-Pipeline	sn.xoffline	0	-	-	-	-	-	-	-	-	-	-	-	-				patrick.sutton	-	Burst SuperNovae
31	R	BURST	Optical supernova triggered - offline (t	Burst SuperNovae			0.1	-	100,000	100,000	-	8,000	24,000	-	1,000	20,000							*See Burst tab*		
32	B	BURST	All-sky long-duration burst - offline, 20	Burst Long Duration	STAMP-AS	allsky.stamp		90,000	270,000	140,000	35,000	130,000	60,000	68,000	400,000	600,000	-	-	-				marieanne.bizouard	0.8	Burst Long Duration
33	B	BURST	All-sky long-duration burst - offline, 10	Burst Long Duration	cWB2G	allsky_ld.cwboffline		-	-	-	-	-	-	-	-	-	-	-	-				sergey.klimenko	-	Burst Long Duration
34	B	BURST	All-sky long-duration burst - offline, 25	Burst Long Duration	X-Pipeline	allsky_ld.xoffline		-	-	-	23,000	61,000	61,000	35,000	152,000	225,000	41,000	128,000	191,000				patrick.sutton	0.5	Burst Long Duration
35	B	BURST	All-sky long-duration burst - offline (tot	Burst Long Duration			****	90,000	270,000	140,000	58,000	191,000	121,000	103,000	552,000	825,000	41,000	128,000	191,000				*See Burst tab*		
36	Y	BURST	EBBH search - offline * 2500 lags	Burst Eccentric BBH	CWB2G	ebbh.cwbof	****	115,000	110,000	110,000	23,000	38,000	76,000	39,000	190,000	308,000	390,000	1,900,000	3,080,000				sergey.klimenko	3.6	Burst Eccentric BBH
37			*Total High Priority*					*320,000*	*590,000*	*460,000*	*108,400*	*316,500*	*338,500*	*190,100*	*1,068,500*	*1,728,500*	*823,000*	*3,959,000*	*6,382,000*						
38	Additional		sources with low detection probability but high scientific payoff, and searches under development for future runs that will be tested in the first observing run.																						
39	RYBG					Unique ID	Storag	Development/testing (SU)			Simulations (SU)			Production (SU)			Additional for detection followup			duction timing rel. to run (mon		latency rel. to (	Contact name		
40	Code	Group	Search	Pie Name	Software	job schedu	Tb/yea	2015	2016	2017	2015	2016	2017	O1	O2	O3	O1	O2	O3	Start	End	taking (hours)	(albert.einstein)		
41	R	BURST	Joint Radio Transients - GW burst (ca	Radio transients	X-Pipeline	jrt.x	0.3	-	-	-	6,000	10,000	15,000	12,000	32,000	48,000	750	16,000	16,000				brennan.hughey	0.079	Radio transients
42	B	BURST	GRB-triggered long-duration burst sea	GRB (long duration bu	STAMP-st	grb.stocktra	****	25,000	-	-	-	-	-	150,000	420,000	630,000	440,000	440,000	440,000				eric.thrane	1.1	GRB (long duration burs
43	B	BURST	IMRI search - offline *	Burst Interm mass ratio	CWB2G	imri.cwboffli	****	115,000	110,000	110,000	23,000	38,000	76,000	26,000	127,000	205,000	347,000	1,455,000	2,182,000				sergey.klimenko	2.6	Burst Interm mass ratio
44	B	BURST	IMBBH search circular constraint - onl	Low latency Burst BBH	CWB2G	imbh.cwbon	20	35,000	33,000	33,000	12,000	18,000	38,000	29,000	122,000	183,000	-	-	-				sergey.klimenko	0.3	Low latency Burst BBH
45	R	BURST	GRB-triggered plateaus search - offlin	GRB plateaus	dedicated	grb_plateau	****	15,000	44,000	22,000	30,000	44,000	22,000	-	22,000	153,000	-	-	-				alessandra.corsi	0.2	GRB plateaus
46	Y	BURST	Hypermassive NS follow-up	SGR/QPOs	cWB2G/Ba	hmns.cwboffline		5,000	5,000	5,000	2,000	2,500	2,500	1,000	1,500	1,500	-	-	-				james.clark	0.0	SGR/QPOs
47	B	BURST	SGR / SGR-QPO triggered searches -	SGR/QPOs	STAMP	sgr_qpo.sta	****	-	-	-	800	400	200	3,000	3,000	3,000	-	-	-				james.clark	0.0	SGR/QPOs
48			*Total Priority*					*195,000*	*192,000*	*170,000*	*73,800*	*112,900*	*153,700*	*221,000*	*727,500*	*1,223,500*	*787,750*	*1,911,000*	*2,638,000*						
49			*Total ALL Priorities*					*1,200,500*	*1,519,000*	*1,217,000*	*610,150*	*1,117,300*	*1,259,000*	*1,219,110*	*4,647,370*	*7,200,450*			*30,189,000*						
50	Future																								
51	RYBG					Unique ID	Storag	Development/testing (SU)			Simulations (SU)			Production (SU)			Additional for detection followup			duction timing rel. to run (mon		latency rel. to (	Contact name		

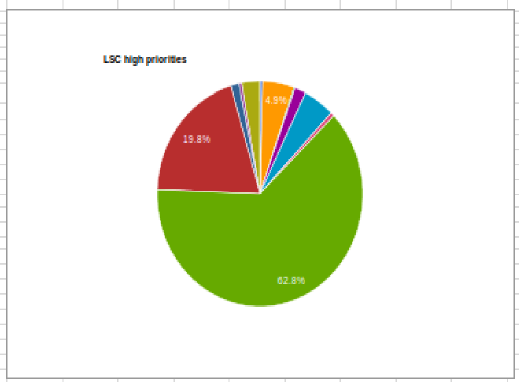
LVCN priorities

	2017 development	2017 Simulations	O3 Production	GW follow up	2017 total
Highest Priority (DAC list 4:50)					
Burst All-sky	0.36	0.27	3.22	19.24	23.08
Burst low latency	0.07	0.04	0.37	0.00	0.47
Burst detection params	0.16	0.36	0.50	1.40	2.42
Burst astro events	0.00	0.10	0.15	0.00	0.25
Cosmic strings	0.00	0.00	0.02	0.53	0.55
BNS	0.04	0.07	1.48	0.00	1.59
NSBH	1.55	3.22	65.68	0.00	70.45
BBH	1.53	3.07	52.96	0.00	57.55
CBC detection params	1.19	1.19	15.94	0.00	18.32
CBC low latency	2.84	5.92	22.68	0.00	31.43
CBC GRBs	0.13	0.18	0.55	0.00	0.87
GR tests	0.11	1.08	3.60	0.00	4.79
Targeted pulsars	0.01	0.00	0.09	0.00	0.10
All-sky isolated stars	3.22	1.61	32.20	0.00	37.03
All-sky E@H	0.02	0.01	0.19	0.00	0.22
Directed X-ray binaries	0.22	0.11	2.23	0.00	2.56
Directed Cas-A	1.31	0.63	12.54	0.00	14.48
Isotropic background	0.01	0.01	0.02	0.00	0.04
Directional stochastic	0.05	0.01	0.04	0.00	0.11
SGWB detection params	0.55	0.11	0.44	0.00	1.10
Detector Characterization	0.09	0.00	1.88	0.00	1.97
High Priority (DAC list 57-88)					
Burst GRB	0.00	0.04	0.26	0.03	0.33
High-energy neutrinos	0.00	0.00	0.01	0.00	0.01
Burst BBH	0.11	0.08	0.31	3.08	3.57
Burst SuperNovae	0.10	0.02	0.02	0.00	0.14
Burst Long Duration	0.14	0.12	0.83	0.19	1.28
Burst Eccentric BBH	0.11	0.08	0.31	3.08	3.57
CBC IMBH	0.01	0.03	0.37	0.00	0.42
Spinning BNS	1.93	4.00	39.87	0.00	45.80
CW directed searches	1.31	0.63	12.54	0.00	14.48
X-ray binaries	0.08	0.04	0.80	0.00	0.92
Known Pulsars	0.03	0.01	0.29	0.00	0.33
SGWB from long transient	0.00	0.00	0.01	0.00	0.01
Detector Characterization	0.33	0.00	1.73	0.00	2.06
Additional Priority (DAC list 92:113)					
Radio transients	0	0.02	0.05	0.02	0.08
GRB (long duration bursts	0	0.00	0.63	0.44	1.07

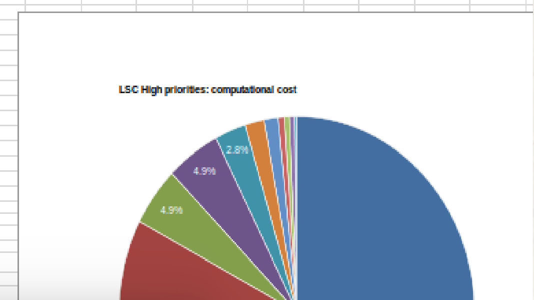


NSBH	70.45
BBH	57.55
All-sky isol	37.03
CBC low la	31.43
Burst All-sk	23.08
CBC detec	18.32
Directed C	14.48
GR tests	4.79
Directed X	2.56
Burst detec	2.42
Detector C	1.97
BNS	1.59
SGWB det	1.10
CBC GRBs	0.87
Cosmic str	0.55
Burst low l	0.47
Burst astro	0.25
All-sky E@	0.22
Directional	0.11
Targeted p	0.10
Isotropic b	0.04





Spinning B	45.80
CW directe	14.48
Burst BBH	3.57
Burst Ecce	3.57
Detector C	2.06
Burst Long	1.28
X-ray binar	0.92
CBC IMBH	0.42
Burst GRB	0.33
Known Pul	0.33
Burst Supe	0.14
SGWB fror	0.01
High-energ	0.01



Compute and storage needs at CNAF for 2016

Taking into account the expected time of observation and engineering runs and the enlarged sensitivity band of the LIGO detectors in 2016 Virgo will need the following amount of computing resources at CNAF:

Cores: 1500 in addition wrt 2015, (total 2500 cores - 25000 Hs06)

Disk: 100 TB in addition wrt 2015, (total approx 600 TB)

Tape: 500 TB in addition wrt 2015

Probably must revisit the HS06 <-> core conversion factor

Uncertainties of compute and storage estimates

Storage estimates are derived from detector bandwidth and expected duration of observation times known in advance, so there not too much uncertainties here.

As for the compute needs the source of uncertainties are as follows:

- Which pipelines will be ported to GPUs and when
- How successfully the sharing of resources will happen with LIGO collaboration
- Possibility of Condor CLOUD installation
- Available budget for computing in EGO
- Currently Virgo is providing only c.c 8% of the total computing power for the LVC collaboration. This cannot be mainteined on the long term, there is a need for massive increase.

The Compute Backend (CB)

The problem

- For several reason (cost efficiency, manpower, future hardwares, etc..) the analysis code has to be generic
- It is always a subject of debate which language to use to program GPUs.
- Double coding for multiple interface is a waste of time and manpower.

The solution:

- THE **COMPUTE BACKEND** (CB) IS ADDRESSING THIS PROBLEM BY PROVIDING UNIFIED INTERFACE FOR VARIOUS GPU PROGRAMING LANGUAGES, SUCH AS CUDA AND OPENCL !
- It levreages the burden of host-side double coding and the very same code can be used to run on CUDA (NVidia) or OpenCL (AMD, Intel, Samsung, etc...) devices...

Compute Backend (CB) features:

- C and C++ API (fortran, python and c# on the way...)
- CUDA and OpenCL backends (ComputeGl, RenderScript considered)
- Single host-side code for multiple backend
- Runs under Linux/Windows/MacOS
- Compatible with CMake, Autoconf, MSVC, etc.
- Academic license is available
- User support around the clock
-

Compute Backend is available on

http://x-perception.com

The Compute Backend - the C API

```c
#include <stdio.h>
#include <stdlib.h>
#include <cb.h>

int main() {

   // Auxiliary variables
   int err ;
   int i;

   // Sets the log level
   cb_log_level = 5;

   // Get some buffer
   unsigned int num_elements = 1024;
   unsigned int size = num_elements * sizeof(float);

   // ... and also on the host side
   float * h_buffer1 = (float *) malloc(size);
   float * h_buffer2 = (float *) malloc(size);
   float * h_buffer3 = (float *) malloc(size);

   // ... fill up the buffers
   for (i = 0; i < num_elements; i++) {h_buffer1[i] = 4; h_buffer2[i] = 11;}

   // The C API
   // A compute backend
   cb_backend   backend;
   cb_program   prog;
   cb_kernel    kernel1, kernel2, kernel3;
   cb_buffer    buffer1, buffer2, buffer3;

   // Get the compute backend
   err = cbGetComputeBackend(&backend);

   // Get a program
   err = cbGetProgram(&backend, "/home/me/testt", &prog);

   // Get the kernel
   err = cbGetKernel(&prog, "test_kernel",    &kernel1);
   err = cbGetKernel(&prog, "simple_kernel",  &kernel2);
   err = cbGetKernel(&prog, "buffer_kernel",  &kernel3);

   err = cbCreateBuffer(&backend, CB_READ_WRITE, size, NULL, &buffer1);
   err = cbCreateBuffer(&backend, CB_READ_WRITE, size, NULL, &buffer2);
   err = cbCreateBuffer(&backend, CB_READ_WRITE, size, NULL, &buffer3);


   // Send some data to device
   err = cbWriteBuffer(&backend.queues[0], &buffer1, size, h_buffer1, true);
   err = cbWriteBuffer(&backend.queues[0], &buffer2, size, h_buffer2, true);

   // Set the kernel sizes
   cbExtent g_size = cbSetExtent(1,1024);
   cbExtent l_size  = cbSetExtent(1, 32);

   // Execute the kernel
   cbParam b1_arg = cbBuffer(&buffer1);
   cbParam b2_arg = cbBuffer(&buffer2);
   cbParam b3_arg = cbBuffer(&buffer3);
   cbParam n_arg   = cbInt(100);

   err = cbExecuteKernel(&backend.queues[0], &kernel3, g_size, l_size, 4,
&b1_arg, &b2_arg, &n_arg, &b3_arg);

   // Read back the result
   err = cbReadBuffer(&backend.queues[0], &buffer3, size, h_buffer3, true);

   // Printing the result
   for (i = 0; i < 10; i++) printf("%f ", h_buffer3[i]);
   printf("\n\n");

   // Releasing stuff
   free(h_buffer1);
   free(h_buffer2);
   free(h_buffer3);

   // Exit
   return err;
}
```

The Compute Backend - the C++ API

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <cb.hpp>

int main() {

    // Sets the log level
    cb_log_level = 5;
    int err ;
    int i;

    // Get some buffer on the host side
    unsigned int num_elements = 1024;
    unsigned int size = num_elements * sizeof(float);

    float * h_buffer1 = new float[num_elements];
    float * h_buffer2 = new float[num_elements];
    float * h_buffer3 = new float[num_elements];

    // ... fill in the buffers
    for (i = 0; i < num_elements; i++) {h_buffer1[i] = 4; h_buffer2[i] = 11;}

    // Construction Backend, Program, Kernel and Buffers
    cb::Backend bck;
    cb::Program prg(bck, "/home/me/test");
    cb::Kernel  TestKernel(prg, "test_kernel");
    cb::Kernel  SimpleKernel(prg, "simple_kernel");
    cb::Kernel  BufferKernel(prg, "buffer_kernel");

    // Initializing the buffers
    cb::Buffer  b1(bck, CB_READ_WRITE, size, NULL);
    cb::Buffer  b2(bck, CB_READ_WRITE, size, NULL);
    cb::Buffer  b3(bck, CB_READ_WRITE, size, NULL);

    // Send data to device
    b1.Write(bck.GetQueue(), h_buffer1);
    b2.Write(bck.GetQueue(), h_buffer2);

    // Set the kernel sizes
    cb::Extent g(num_elements);
    cb::Extent l(32);
```

```
   // Create kernel arguments
    cbParam buff1_arg = cbBuffer(b1);
    cbParam buff2_arg = cbBuffer(b2);
    cbParam buff3_arg = cbBuffer(b3);
    cbParam numarg    = cbInt(100);

    // Execute the buffer kernel
    BufferKernel(bck.GetQueue(), g, l, 4, &buff1_arg, &buff2_arg, &numarg, &buff3_arg);

    // Read back the result
    b3.Read(bck.GetQueue(), h_buffer3);

    // Some output for checking the result
    for (int i = 0; i < 10; i++) {
        std::cout << h_buffer1[i] << " " <<  h_buffer2[i] << " " <<  h_buffer3[i];
    }

    // Releasing stuff
    delete h_buffer1;
    delete h_buffer2;
    delete h_buffer3;

    // Exiting
    exit(0);
}
```

Compile for CUDA:

```
cd build
cmake -DOPENCL_BACKEND=1 ../
make
```

Compile for OpenCL:

```
cd build
cmake -DCUDA_BACKEND=1 ../
make
```

Problems

- Graphical monitoring of user jobs
- Interactive debugging, checking logs
- Account request procedure at CNAF is inconvenient for LIGO colleagues. Maybe could be simplified with a certificate signed request instead of faxing a sheet ?

Good things

- Resources reliable, negligible downtime
- Support mailing list (CTCC@EGO) and CNAF user support works fine ! (Thanks !)