

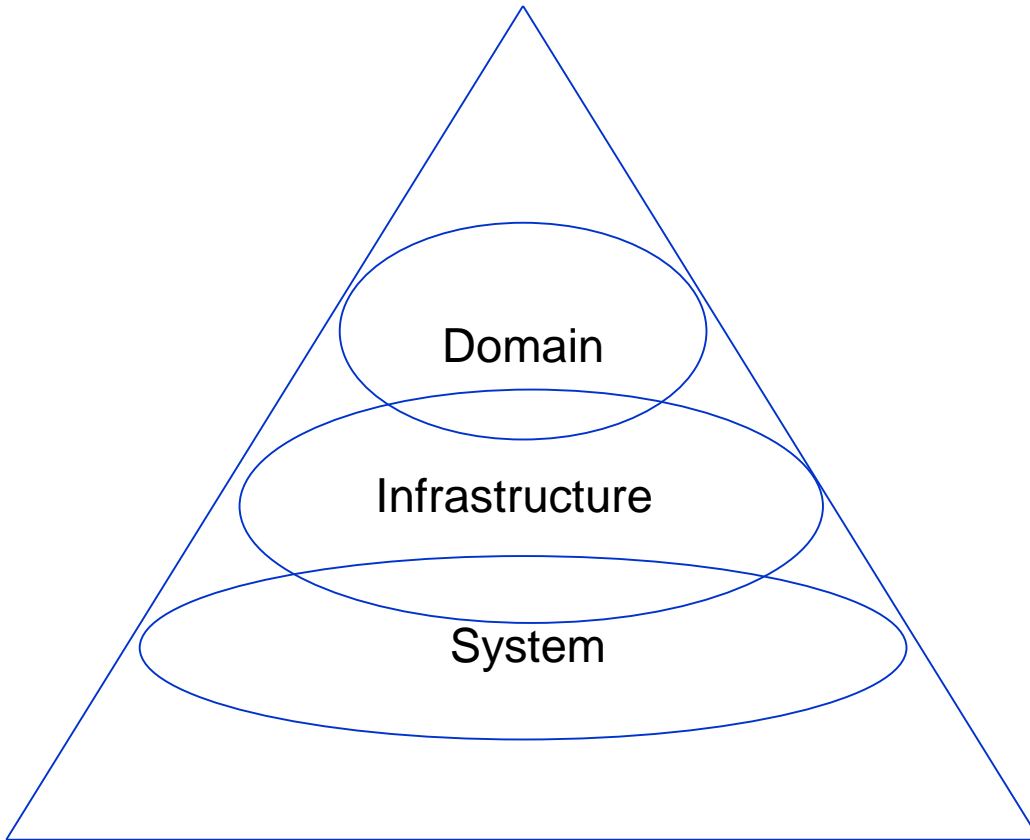
Ste Jones  
University of Liverpool  
GridPP  
[sjones@hep.ph.liv.ac.uk](mailto:sjones@hep.ph.liv.ac.uk)

- Liverpool is a T2 grid site, part of the GridPP collaboration and working within WLCG.
- We have ~ 2450 slots (~25KHS06), about 1,700 are HTCondor (others are VAC)
- Two HTCondor instances; one SL6, the other CentOS7. Separate ARC CEs.
- Using HTCondor since ~ 2014; migrated from PBS.
- DPM for storage (1.5 PB)

- HTCondor is a product, but it is not (necessarily) an application. Like operating systems, networks, database management systems, and security infrastructures, HTCondor is a general system, upon which other applications may be built.
- Extra work is often needed to create something useful from HTCondor. The extra work depends on the goals of the designer. This talk identifies a few general areas that need to be addressed and gives specific ways that they were actually solved when adapting HTCondor to work in the grid environment.

- Fours Ps
  - Product
  - Project
  - People
  - Process

- This talk considers how the infrastructure can be adapted to suit the domain or vice versa (graphic courtesy of J Coles).



| Requires knowledge of:   | Example daily tasks:  | Additional activities   |
|--|---|---|
| Experiment workflows;  | Support users with job submission;  | Study and support performance improvements;   |
| Distributed computing (grid; cloud; federated storage);                | Run site/national service nodes; research emerging technologies and techniques. | Test new releases; experiment with configuration extensions, enhancements and improvements; |
| Compute hardware; machine room dynamics; networking; Operating systems | Install new hardware; patch worker node OS; run monitoring                      | Tender for hardware; upgrade machine room; test new server types;                           |

- Research and development
  - Feasibility research on the usefulness of “good” technologies and trends.
    - Please don’t ask me what “good” means – it would take all day. But I know it when I see it.
  - Middleware integration research to test and integrate potentially useful technologies into production.
    - Much of the R&D for the methods used to integrate HTCondor into the grid were done by A Lahiff while at RAL.)

- System reliability DevOps!!!)
  - Build and version control automation
  - Standards, documentation, reuse.
  - Backup, recovery and redundancy of systems.
  - Continuous migration, at various levels:
    - Hardware
    - Operating systems
    - Applications
    - Middleware and support software

- **User/Experiment Support**
  - VO On-boarding and support
    - This is an essential project management/communication task for all new VOs. It involves at least the following sub-steps.
    - Identify organisation stakeholders.
    - Identify organisation requirements and rationalise.
    - Design an initial baseline for organisation.
    - Obtain support at a subset of UK sites, and enable compute/storage.
    - Training for the organisation (esp. tools, certs, proxies and VO set-up or enrolment steps, job design and control etc.)
    - Small scale tests to check feasibility.
    - Large scale roll-out, and continuing operations.

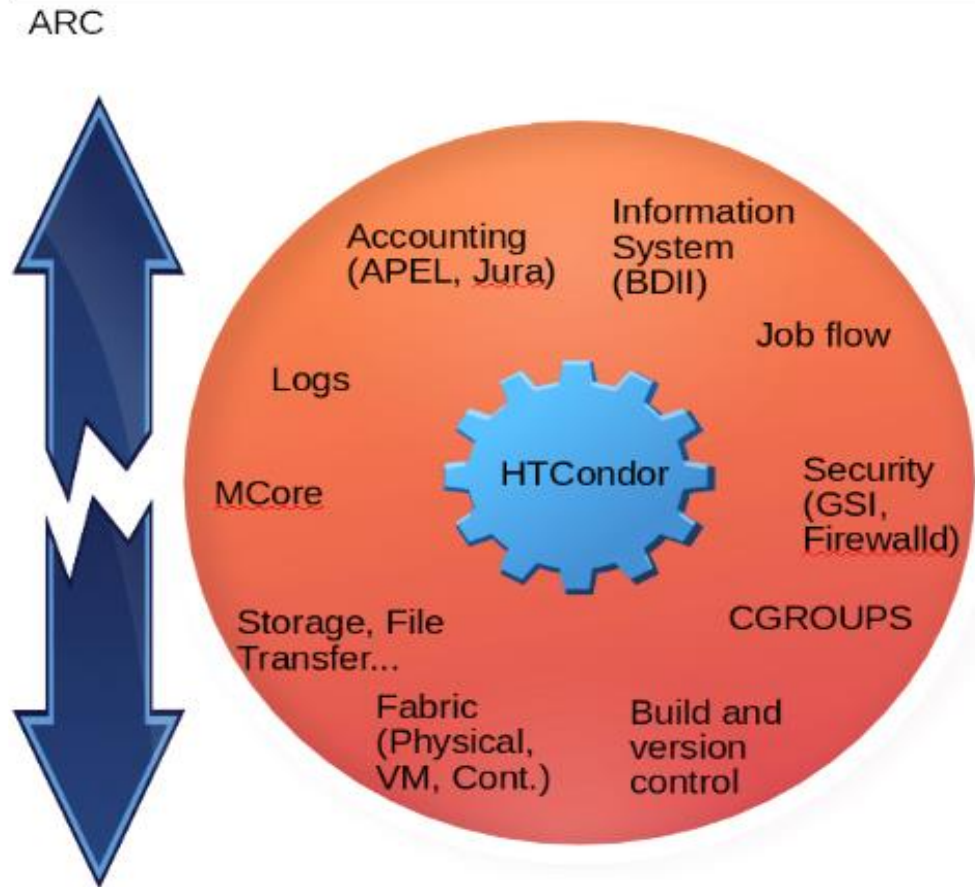




- All roles
  - The background skills for all these rolls are listed above on slide 2. In addition, general useful skills should include quality assurance/CMM, general IT, science or engineering skills, knowledge of Open Source culture, general knowledge of tools (yum, rpms, puppet, ansible), the soft skills, and an inquiring mind. For more specialised roles:
- Research Infrastructure Engineer
  - This involves fabric provision. Therefore, this role would call for particularly solid design, installation and configuration expertise and knowledge in hardware (and on-going maintenance), as well as advanced knowledge of networks and storage technologies, e.g. RAID6, or ZFS etc.
- Research S/W Engineer
  - This would call for particularly solid programming skills (bash, Perl, Python, Java, C, SQL .. ), current concepts (Web Services, Databases, networks), knowledge of the e-infrastructure ecosystem, and strong knowledge of current and emerging Grid frameworks (GSI, storage protocols, Cloud, commercial provision, middleware concepts and software internals.) Massive Physics expertise required for development of modelling algorithms.
- Site Reliability Engineer (DevOps?)
  - This is “what happens when a software engineer is tasked with what used to be called operations.” It involves strong debugging and workaround skills, decent knowledge of application and OS internals, and total familiarity with automation tools such as Puppet/Hiera or Ansible. According to Wikipedia, “a site reliability engineer (SRE) will spend up to 50% of their time doing “ops” related work such as issues, on-call, and manual intervention. Since the software system that an SRE oversees is expected to be highly automatic and self-healing, the SRE should spend the other 50% of their time on development tasks such as new features, scaling or automation. The ideal SRE candidate is a highly skilled system administrator with knowledge of code and automation.” I’d add QA, PM and S/W Eng skills to that list.



## Integration aspects



Roll your own

# Chunks of integration

- Additional products exist that provide “chunks of integration” to the HTCondor system. E.g. Nordugrid ARC is one of them.
- Some of the “chunks” are complete. Others just give sockets to plug-in your own work.
- Even ARC does not cover it all. Sometimes you just have to roll your own.
- This may mean building a bridge to some other product (impedance matching?) or creating a specific product to realise the functionality.

# Chunks of integration

- HTCondor itself, but entire subsystems exist external to HTCondor.
- BDII
- Accounting
- ARGUS security
- Various submission APIs
- Storage systems
- Build/config control standards and systems
- GOCDB
- VOMS Portal/VOMS records management and dissemination
- Tickets (bugs, problem reports)
- Testing and monitoring systems.

# Chunks of integration

- I'll briefly touch on a few aspects of this work that was done to integrate HTCondor into a Grid environment at Liverpool. Not all of it; there's too much for 20 minutes.
- Much of the work is described in painful detail in the general refs section at the end.
- I'll try to give meaningful references wherever appropriate.

- ARC provides an accounting application, JURA, that surfs through the HTCondor job logs, prepares accounting logs in the appropriate format and transmits them into the APEL grid accounting database via the Secure Stomp Messaging (SSM) queues.
- That's a good “chunk of integration”, and it saves a lot of routine coding. But it's not enough to finish the job.
- We use workers of varied strengths (HS06). Depending where a job lands, more or less work is done over a known period. Without “doing something” APEL (in the way we use it) wouldn't be aware of that variation. Accounting would be wrong.
- What we do is “scale” (i.e. alter in the log file) the time the job runs for proportionately to the strength of the workernode.
- A job on a weak node is scaled to shorter time & vice versa. The node has a knowledge of its own power per slot. This is held in a Custom Attribute (RalScaling) that is passed as a parameter to an an ARC “authplugin” script, `scaling_factors_plugin.py`, that gets run when a job finishes.
- Also developed a scripted way to republish accounting records when dataflow/storage goes wrong.
- See Accounting and General sections of refs.



- A good deal of effort was put in to supporting jobs which require more than one queue (typically 8 cores).
- It is almost impossible for 8 single slots to become available at the same time without “doing something”. A variety of measures may be applied, usually holding slots empty for a while (i.e. draining) until sufficient have drained to make an “8 core slot”, and prioritising jobs with more than one core so that they run first (i.e. to “grab” the slot!)
- Much of the functionality is available in core HTCondor, but we (i.e. Liverpool and some others) use home-grown “shim” software (Fallow) to orchestrate the draining to make it proportional to some ideal set point. Raises the ceiling when no MC jobs queued, stopping needless idletime. Better to drain larger machines (RALPP idea.)
- See Multicore section of refs. The “Barcelona” reference, in particular, spells out how this part of the system can be configured.
- We make use of GROUPs and subgroups. Users of MSCORE jobs are given a “good” priority factor (a very one), to ensure MSCORE slots are used up first. We then control the number of MSCORE slots by draining a proportionate amount.



- GSI (Grid Security Infrastructure)
- Highly integrated into the suite via (from the bottom up) Linux, HTCondor, ARC, Icas/Icmaps, ARGUS.
- Incorporating the VOMS system, e.g. VOs, CAs, certs, proxies, authn/authz, account mapping, central banning, logging, revocation.
- Uses a Plugin approach in ARC to make the scheme generic.
- Largely transparent at the HTCondor level.
- Uses mass user account approach, which is being dumped in favour of containers nowadays.
- See General refs section for some info on this, but there's no "great" reference for it, so I'll lay it out the flow in a bit more detail over the next slides.





- An incoming job has a proxy attached to it. ARC has a “plugin point” (called a unixmap in arc.conf) which is used to call out to some third party scheme when a proxy needs to be checked and used to map a user to a local user account.
- The unixmap specifies a call to the lcms subsystem (a piece of middleware from glite).
- The lcms subsystem is itself configured to make a call to some ARGUS server. ARGUS can take a proxy, compare it with some policies, and authenticate the user, returning a local username for the user to be “mapped to”.
- If the answer is no, ARC drops the job and gets on with its life. Else it changes the account of the user and submits the job into the batch system HTCondor in this case.
- You’ll find the setup for this in the General section refs.



- This is about integration with the OS. We've written an article to summarise how HTCondor can operate with the Linux CGROUPS architecture to limit the resources a job can take:
  - [https://www.gridpp.ac.uk/wiki/Enable\\_Cgroups\\_in\\_HTCondor](https://www.gridpp.ac.uk/wiki/Enable_Cgroups_in_HTCondor)
- This config imposes "soft limits". With this in place, the job is allowed to go over the limit if there is free memory available on the system. Only when there is contention between other jobs for physical memory will the system force physical memory into swap. If the job exceeds both its the physical memory and swap space allotment, the job will be killed by the Linux Out-of-Memory killer.

- We've integrated the HTCondor system within an infrastructure for automatically building the head and worker nodes and all the associated extensions (ARGUS, BDII, APEL).
- When “grid options” were narrower, the YAIM tool set the standard for the integration of grid tools and layers.
- As far as I know, YAIM never had complete support for an ARC/HTCondor setup, but it was still useful for many of the configuration tasks, such as VOMS, making users.
- Now grid options are “wide”, and there is no standard grid site.
- In the recent move from SL6 to CentOS7 at Liverpool, we found that YAIM was becoming ever more useless (although it is still recommended for the BDII, I believe).



- Consequently, since there is no central support for YAIM, vendors and users have started to create specific automation tools for their products. A typical example is DPM which is in widespread use. Puppet is the de facto standard.
- “People” have created various similar solutions for HTCondor and/or ARC. But an agreed, single de-facto standard is yet to emerge. I direct those who are interested to see the talks at GridPP40 at Pitlochry (see refs).
- Luke Krezko (see his talk, earlier) and Alessandra Forti of GridPP have taken big steps towards standard Puppet/Hiera modules for this task. Puppet modules cover a given configuration area, Hiera hierarchies parameterise the modules (i.e. provide variability).
- But Liverpool has “rolled its own” for the task, producing mostly site specific puppet modules that entirely eliminate YAIM from the build. This was done for “pragmatic” reasons. We will monitor the situation and perhaps adopt Luke’s standard.



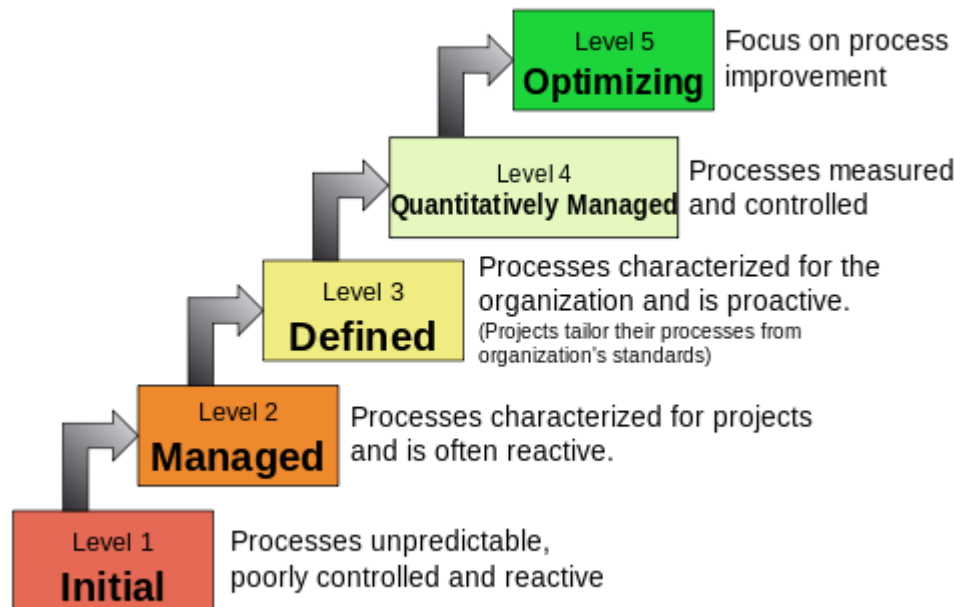
- Recent advice (on the TB\_SUPPORT mail list) is:
  - Your starting point for puppetising the grid config should probably be the puppet-voms (<https://github.com/cernops/puppet-voms>) and puppet-fetchcrl (<https://github.com/voxpupuli/puppet-fetchcrl>) modules (or grab the UK forks from <https://github.com/HEP-Puppet>) which provided basic functionality which underpins almost all node types.
  - There are more modules in the HEP-puppet github area, you can either fork them or use them for ideas.
- This is not good “component oriented design” We need to do much work to make this more seamless.

# Future Goals

- Until now, batch systems with CE front ends have been the work horse of grid computing for the WLCG. New ideas, technologies and possibilities are cropping up all the time. Standards to promote reuse have not kept up with the pace of work. When new things have come along, we've jammed them in to assess their feasibility and get them into production if they pass muster.
- This is, arguably, counter productive. Less reuse. We may need to try to mature as a group at some stage and consolidate and harden up all we have won. That's an overarching goal, and the most pressing matter is probably the state of automation and documentation for moving on to CentOS7.

- I'd estimate that we "Griddies" at T2s are mostly on or about levels 2 and 3 (managed and/or defined) in the Capability Maturity Model (CMM), with some work at level 4 and 5. This is certainly much better than 1 (initial, formerly ad hoc).
- If we go too far, processes become just too rigid. But I think we can get a bit more out of it.

## Characteristics of the Maturity levels



- The “job” is no longer just classic Linux sysadmin. It’s become a research role, involving Component Based Software Engineering (CBSE) and DevOps etc. where the batch system, HTCondor in this case, is used as a platform at the base of an array of maturing components.
- We are taking steps to mature the process, but we disseminate the standards in a “social” way, without formality and with great flexibility.
- This has had “good” results (they found the Higgs), but we may need beef up the standards to get further up the CMM.
- I’ve laid out (please see the refs) some of the areas where we have done work, but there are still many challenges in the areas of standards, documentation, information dissemination; basically, problems of Component Oriented Software Engineering.



- General
  - [ARC Condor\\_Cluster \(SL6\)](#)
  - [Enable\\_Cgroups\\_in\\_HTCondor](#)
- Accounting:
  - [Accounting, Scaling and Publishing](#)
  - [Benchmarking procedure](#)
  - [Publishing tutorial](#)
- Multicore
  - [Defragmentation](#)
  - [Barcelona presentation](#)



- ARC/HTCondor C7 talks at Pitlochry (GridPP40)
  - [CentOS7/SL7 Upgrade issues](#)
  - [Upgrading to SL7/ CentOS7 at Liverpool](#)
- Liverpool's approach to CentOS7
  - [Centos7 Adoption \(Puppet and Hiera\)](#)

- [https://www.gridpp.ac.uk/wiki/Example\\_Build\\_of\\_an\\_ARC/Condor\\_Cluster](https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster)
- [https://www.gridpp.ac.uk/wiki/Enable\\_Cgroups\\_in\\_HTCondor](https://www.gridpp.ac.uk/wiki/Enable_Cgroups_in_HTCondor)
- [https://www.gridpp.ac.uk/wiki/Example\\_Build\\_of\\_an\\_ARC/Condor\\_Cluster#Notes\\_on\\_Accounting.2C\\_Scaling\\_and\\_Publishing](https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster#Notes_on_Accounting.2C_Scaling_and_Publishing)
- [https://www.gridpp.ac.uk/wiki/Benchmarking\\_procedure](https://www.gridpp.ac.uk/wiki/Benchmarking_procedure)
- [https://www.gridpp.ac.uk/wiki/Publishing\\_tutorial](https://www.gridpp.ac.uk/wiki/Publishing_tutorial)
- [https://www.gridpp.ac.uk/wiki/Example\\_Build\\_of\\_an\\_ARC/Condor\\_Cluster#Defragmentation\\_for\\_multicore\\_jobs](https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster#Defragmentation_for_multicore_jobs)
- <https://indico.cern.ch/event/467075/contributions/1143835/>
- <https://indico.cern.ch/event/684659/contributions/2897404/>
- <https://indico.cern.ch/event/684659/contributions/2885521/>
- [https://www.gridpp.ac.uk/wiki/Centos7\\_Adoption](https://www.gridpp.ac.uk/wiki/Centos7_Adoption)

