Improving efficiency of analysis jobs in CMS.

Authors: Stefano Belforte, Matthias Wolf, Todor Trendafilov Ivanov, Marco Mascheroni, Antonio Perez-Calero Yzquierdo, James Letts, Justas Balcas, Anna Elizabeth Woodard, Brian Paul Bockelman, Diego Davila Foyo, Diego Ciangottini, Jose Hernandez Calama, Leonardo Cristella

July 10, 2018

CRAB & Global Pool & glideinWMS

– At most one slide –

Concept

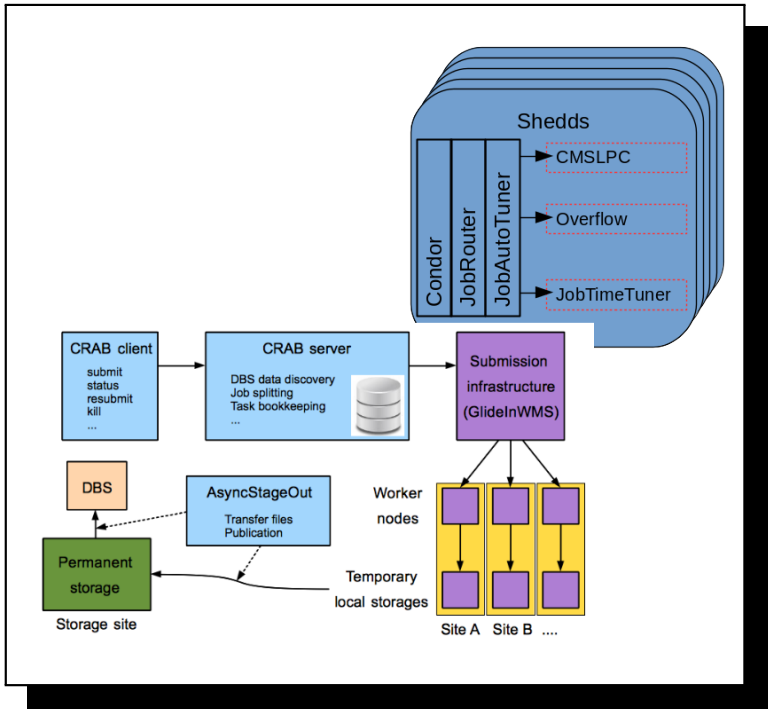
– One or 2 slides here –

JobAutoTuner and CRAB3 Workflow:

What ever we do we do it at the final step, just before the job submission and after the DAG Expansion - We have to be really fast:

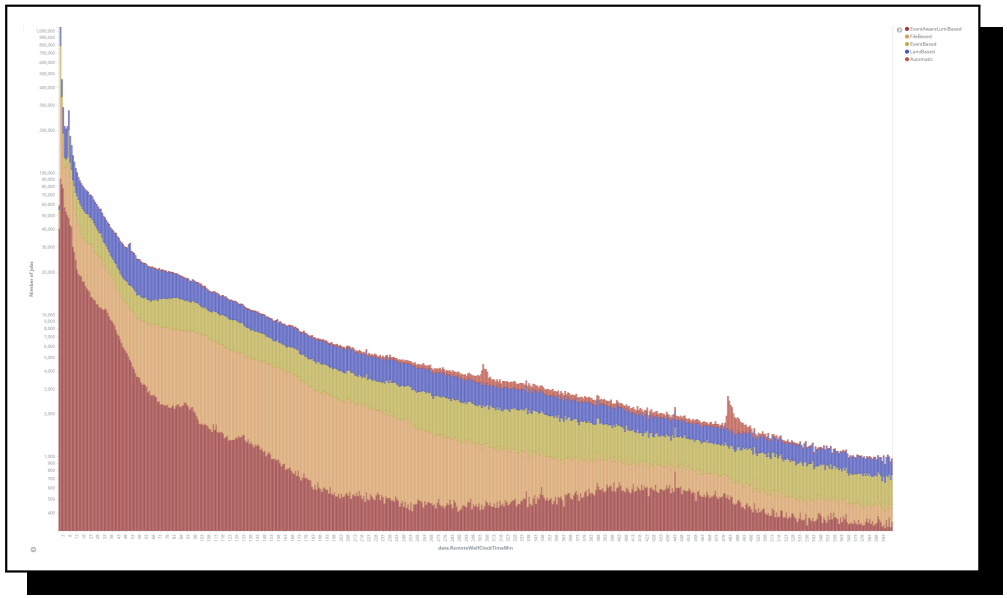
- Fast Script execution.
- Fast in gathering of the Condor related information.
- Fast in querying the external information.
- Fast in taking the decisions.
- Fast in applying it - editing the jobs.

Alternative solution - keep jobs in hold while measuring (Potential enormous delay in submission time).



Concept && differences from Automatic Splitting

– one slide –
– + showing how EWT perJOB progress in time - well defined task + pathological cases. –
– this one may simply be united with the previous slide –
– the image in the current slide is just for scale testing.



Current implementation of Time tuning

– one slide explaining the whole machinery –

Current results

– Few plots showing the results from the latest implementation –

New Methods for improving the accuracy of the current estimation

– One slide showing that there is great motivation and opportunity to apply multivariate analysis and ML for improving the accuracy of the method. –
– And that the proper hooks in the current code are already prepared for such next step. (we may even mention that this is a work in progress or a field yet to be explored) –

Problem:

- **The primary need:** to achieve a better resource utilisation
- **The secondary need:** to protect the sites from being flooded with jobs they cannot process || serve data for them.
- **The old Overflow mechanism** - what does it suffer from:
  - Statically defined overflow regions - can't be based on other criteria characterizing "proximity"
  - Overflow matching decision happens in the timescale of pilot lifetimes - not flexible enough to respond to faster changes in the status of the distributed CPU and storage resources
  - Requires additional FE groups to be set - a limitation in practice to the different number of settings that could be configured at once.
  - Based on a special type of pilots - fragmentation of the resources, increasing wastage

Proposed Solution:

A solution based on the condor JobRouter mechanism to edit jobs in place.
Challenges:

- The difficulties for the estimation - the complexity in putting the right algorithms: What do we want to overflow & Where do we want to send it.
  Sysadmin on a military drill.
- The difficulties of measuring:
  Hard to find the proper metric which best describes one's needs and the effect of his actions.
  one possibility: IO waits accessible directly from the UserLog @ the Schedd
- The complexity of applying all the decisions (separately/together)
- Decentralizing it adds complexity to the communication between the entities and increases the time for achieving coherency.
- In the new CMS computing model **Bandwidth & IO** should be treated as a separate resource in addition to **CPU & Storage**

History - TimeTuner:

– This slide must be revisited and if there is something left out of it to be shifted in the Time Tuning section –
The first attempt to exploit the CondorJobRouter machinery.
It definitely works fine - it scales and is doing what it is expected to do.
What kind of problems did we face:

- Proper communication with the external systems.
- Time delays in the returned information.
- Correct monitoring.
- Efficiency.
- To protect the JobRouter.
- To protect the Collector - we basically did not go out of the schedds themselves.

What did we learn:

- One cannot solve a problem which does not exist.
- One cannot measure the world while looking at his internal load.

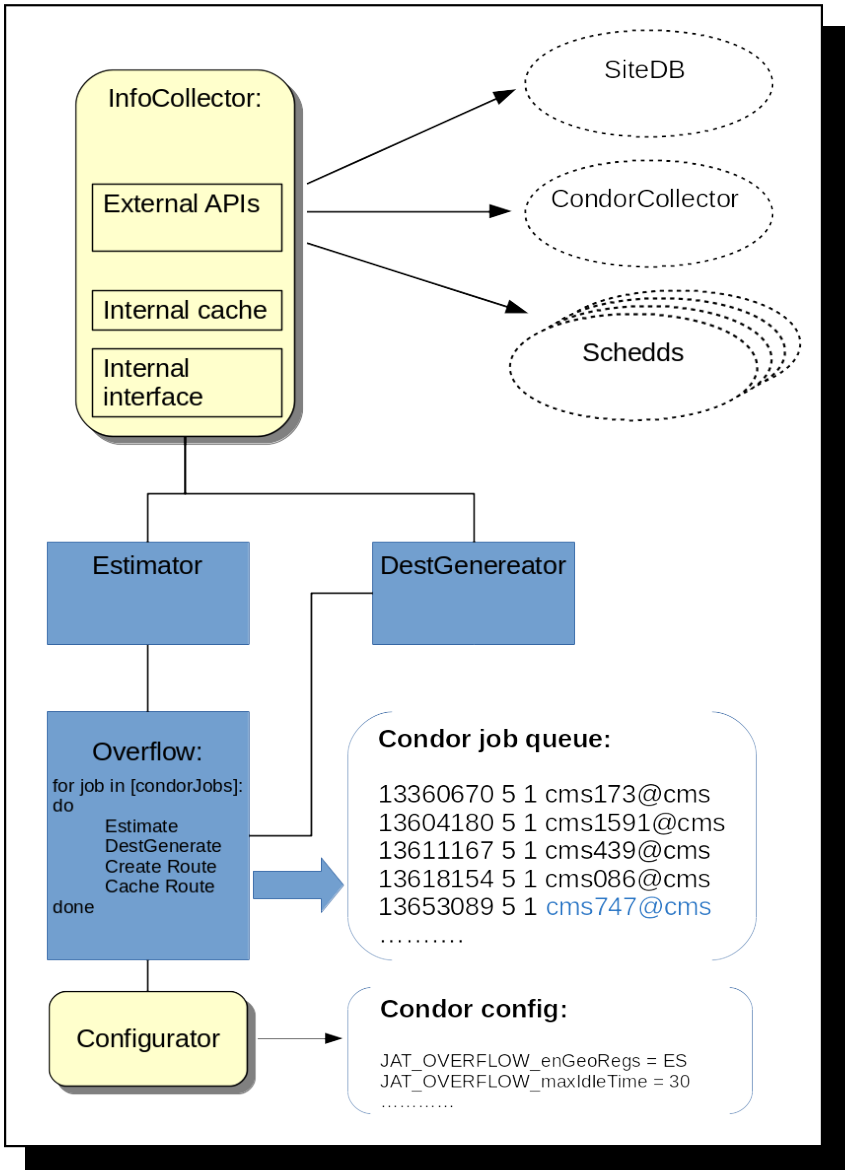
We came to the current sate step by step.

- Implementing a maximum overflow in a country and providing a way to substitute the old Overflow.
- Facing the Tier1 problem - Critical for us. In the current situation of the negotiation order Analysis jobs cannot run on the Tier1 sites but there are datasets that are currently placed only at T1.

Structure:

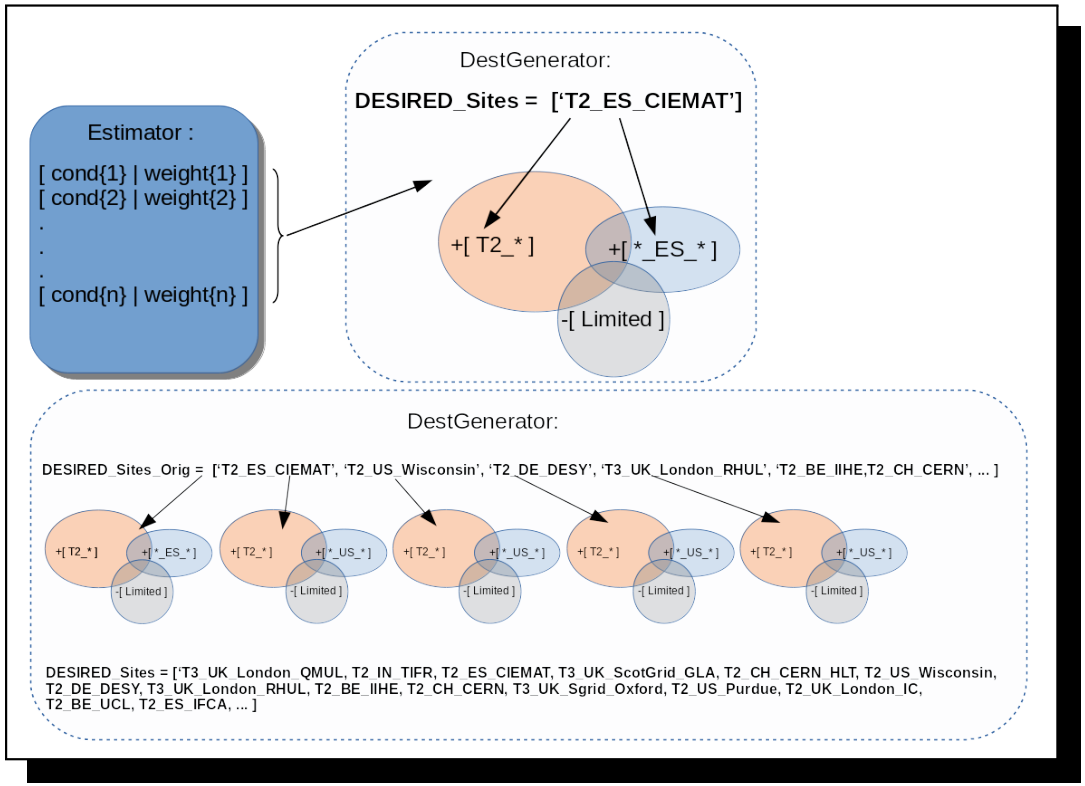
Three basic abstractions:

- **Information Lifetime:**
  - static
  - dynamic
- **The OverflowLevel:**
  - PERTASK
  - PERJOB
  - PERBLOCK
  - PERFILE
  - PERDATASET
- **The OverflowType:**
  - GEO
  - TIER1
  - TIER2
  - DATALOCATION
  - LOCALLOAD
  - SRCLOAD
  - DSTLOAD



Decision making & Subsets

- Weighted sum vs. Weighted single decision.
- Estimating the weights could be dynamic:
  In the future we can apply more elaborate mechanisms for estimating the optimal weights according to the prompt feedback about the reaction of the system.
- Subsets intersections.



Limits & Protect ourselves

Limitations - where and how can be implemented:
We do not want to shoot ourselves in the thumb - so we need to have more options than just maximum overflow.
Alternatives:

- **At the Schedd level** - Inside the script:
  Implemented but we cannot say how much can we scale with that.
  - **Pros:**
    We will have the system solid and configured at one place.
    The configuration limits are going to be accessible through the script itself.
  - **Cons:**
    It requires addition queries to the collector (schedds).
    Additional delays.
    Additional memory for the caching system.
- **At the Negotiator level** - Concurrency limits.
  - **Pros:**
    Fast solution. Not requiring additional coding.
  - **Cons:**
    We will have the system configured in many places.
    Need additional cycle during the negotiation process - creates more pressure on it.
    We will brake the possibility to propagate the feedback information (we will not be able to recognise if we hit the limit set in the negotiator or some capacity limit).

Current Status:

- **What is already done:**
  - **Information Lifetime:** The caching system of the script is working well.
  - **The OverflowLevel:**
    - PERTASK - implemented & tested
    - PERJOB - implemented & under testing
    - PERBLOCK - R&D
    - PERFILE - R&D
    - PERDATASET - R&D
  - **The OverflowType:**
    - GEO - implemented & tested - (Default)
    - TIER1 - implemented & under testing
    - TIER2 - to be discussed
    - DATALOCATION - R&D
    - LOCALLOAD - to be discussed
    - SRCLOAD - R&D
    - DSTLOAD - R&D
- **What is left to be done (near future):**
  - A proper monitoring and a scale test.
  - Moving it to a daemon.
  - Finalising the rest of the OverflowLevels.
  - Refactoring the Destination Generation in order to better tie it to the way we do the decision making.
- **What can be done in the future:**
  - To make it possible to measure the actual situation (load / resource utilisation) at the sites and mechanism for propagating the feedback.
  - To have more topology aware information accessible.
  - A hard one: to make it more general and context independent.

– to list the future plan for improving the current implementation (if we decide to have a section with 'future plans') –
– to mention Automatic Memory Tuning as a candidate area for future research –