Applying Distributed HPC CyberInfrastructure to Sequestrate C02

Shantenu Jha[†‡§], Yaakoub El Khamra[*†]

[†] Center for Computation and Technology, Louisiana State University, Baton Rouge, 70803

[‡] Department of Computer Science, Louisiana State University, Baton Rouge, 70803

[§] e-Science Institute, Edinburgh, UK

[*] Texas Advanced Computing Center, University of Texas, Austin, 78758

***Abstract*—Here I will put broad strokes for now some ideas for what we should include in the abstract: We will also come up with a better title..**

***Index Terms*—Distributed and Autonomic Applications, Distributed Application Programming, SAGA**

NOTES

- ***Jha: Define Problem in a few sentences; Motivate/Establish the Use of Distributed CyberInfrastructure
- Sensor driven application and workflow: the number of stages is controlled by sensors and the simulations themselves use sensor data
- Highly unpredictable: we have workflows that are changing dynamically and applications that use parameters from live data that we do not see before, everything will vary wildly
- The sensor data can be quite critical: if you are injecting CO2 and suddenly (i.e. real-time suddenly) you lose well-head pressure: the CO2 is going somewhere, you want to find out where! It could be building up around the casing or going into a fracture, really nasty stuff can happen
  - Based on that, we will need several things: frequent and dynamic updates of available sensor data (i.e. sensor platform, which we sort of have but needs more work, this is labview side...)
  - On demand computing: You want to know if the CO2 is just moving to new places normally (i.e. you have low permeability and have to cross a threshold to get CO2 through there) or you broke the rock and CO2 is building around the casing of the well and the whole damn thing will explode
  - Special handling of critical sensor data: we might need some sort of mechanism to distinguish low priority sensor data from critical sensor data, based on that, throw a token for spruce or switch to dedicated queues or something like that
  - The data is sensitive and confidential, security is a big issue that we will need to address, same goes for database redundancy (you don't want all ensembles hitting a single DB for latest sensor data all at once, you want redundancy...)

  - Based on sensor data, we might want to run fire control models, emergency response models, kill-well/well-shutdown models etc... So analysis of sensor data before we jam it in the workflow might be important

- While the application in this case is CO2 seq, this can be really anything in the real world EnKF for atmospherics, EnKF for radar, or simply hurricane simulations
- Keeping tabs, book-keeping is incredibly difficult, you will need to archive data effectively for larger runs (list this in the challenges)
- In terms of autonomics, we have: sensor driven workflow (i.e. throughput), sensor driven simulations, i.e. data-aware workflows and data-aware simulations. We do not have this now but a natural extension is checking whether the data is critical or not, this would make the application data-criticality aware (not just availability of data but nature of the data... need better term for that one) ***Jha: this needs a bit of clarification
- We have (or will have soon once the admins fix it) condor, that should help with the on-demand computing issue by allowing us to harvest cycles for regular jobs and put the critical ones in the high-priority queues
- One interesting feature: when we run this we are doing 2 things: history match then forecast, i.e. we are updating our "study" of the reservoir behavior dynamically as it evolves in realtime
- Need to formulate the problem in terms of Application Objective, Mechanisms and Strategy.
- The three Unique things about this paper are (i) The fact that we are studying C02 Sequestration using EnKF (EnKF-CS) one of the first to do so, (ii) Effective/proper solution of EnKF-CS, imposes interesting additional requirements, over and above EnKF for History Matching – that of sensors & experimentally driven data. We incorporate this additional requirement, using the same programming system (SAGA) as used for EnkF-HM thus justifying claims of extensibility, and (iii) Due to distribution of sub-problems (task), we introduce Condor pools in the mix of resources we utilize and interestingly, we utilize Condor's native pilot-job (Glide-In) when using Condor resources but SAGA's BigJob abstraction when using TG resources (which opens the question, why can't we use Condor's Pilot-Job on Ranger, QueenBee ?). We beleieve this is a novel if not the first demonstration of the use of two-different pilot job mechanims towards the solution of the same problem.