Avoiding complexity in the development of corporate grid applications using the REST api

Timo Strunk
Nanomatch GmbH
Talk

• Our UNICORE Use-case
  – Why we require UNICORE

• Our Software until now
  – Difficulties
  – What went wrong, what did we learn?

• Current and Future implementations
  – REST API
  – UI
About Nanomatch

- Spin-off company based on Code of MMM@HPC and the Wenzel group in KIT
- We investigate
  - Thin-Film morphologies for OLED based devices
  - Electron/Hole Mobilities
- Or to put this in another way
  - We try to answer
    - Will my TV turn on with a specific material?
    - Will my TV work for more than five minutes?
\[ k_{if} = \frac{2\pi}{\hbar} |J_{if}|^2 \frac{1}{\sqrt{4\pi \lambda_{ij} k_b T}} \exp \left( -\frac{(\lambda_{ij} + \Delta E_{ij})^2}{4\lambda_{ij} k_b T} \right) \]

- Energetic disorder: $\Delta E_{if}$ and $\sigma(\Delta E)$
- Electronic coupling: $J_{if}$
- Reorganization energy: $\lambda_{if}$
\[ k_{if} = \frac{2\pi}{\hbar} \left| J_{if} \right|^2 \frac{1}{\sqrt{4\pi \lambda_{ij} k_b T}} \exp \left( - \frac{(\lambda_{ij} + \Delta E_{ij})^2}{4\lambda_{ij} k_b T} \right) \]

Energetic disorder
Electronic coupling
Reorganization energy
Software

What we wanted

- UI - Click to work
- Input molecule
  - Get Morphology
- Input Morphology
  - Get Mobility

What we had

- Complex
- Script based
  - Parameters in scripts
  - Parameters as commandline
- Some MPI programs
- Some OpenMP

→ We required a unified UI, but needed to interface directly with a cluster / grid
Other implementations
Other implementations cont.
Competitors' solutions

- Specific to grid interface
  - Torque, SLURM, etc.
- ssh based
- Sometimes expensive
- Mostly modular

→ UNICORE
Parameterizer

- First success story
  - Unbundling allows for different allocations
    - Geo Opt
    - Point Charges
- Batch processing
- Negative Points
  - Data duplication
  - Addressed in WF Server 7.x
5M morphology

Shredder

Stage-In: 2.5GB

500 iterations

Input
5M*500 = 2.5GB

7000 jobs
1.5TB of stage-in and out

500 iterations, roughly 10 GB stage in and out

• Stage-in and out dominate runtime
• Sheer amount of jobs not viable for an actual application

→ We were not smart designing this
Single Application UIs

- Scientific origin of our software leads to a large and diverse knowledge of our developers

- Most of us know Python
- Some know C++
- None know Java

- Mix of SWT and Swing
- Not everyone respects the Gridbean-model

Solution
Generic Gridbean
Summary of our mistakes

- Workflows
  - Data transfer often abused
  - Non-portable, not inheritable
- Individual application GUIs
  - Gridbean concept not sustainable in our group
  - Java code ends up non-maintainable due to
    - SWT, Swing mix
    - Gridbean / non Gridbean storage
  - Very slow development cycle for trivial UIs
Lessons learned

- Do not convert your existing applications into Workflows
- Do not invest into GUIs, where none are required (Generic Gridbean)
Current and Future Developments

- REST API allows fast new client development
- Remove development stress from the scientific developers
  - Write Input/Output specification
  - Complex purely declarative GUI akin to Generic Gridbean
- Write multiple input formats
  - No Shell variables, but hierarchical
    - YML, XML
Once you “render” the job, you get a YML with the exact same structure as above minus the markup information.
Workflow encapsulation

- Linear nature of declarative application GUIs allow for workflow "concatenation"
- Workflows can be encapsulated and become new gridbeans
Conclusions

• Past
  – Large freedom in development
  – Bad Choices, long development times
    • Huge Appreciation for Generic Gridbean
  – Custom GUIs only necessary for file preparation
    • Better handled by external thread

• Present and Future
  – Make a universal GUI to easily pass all parameters required for runtime only
  – Do not require any executable code for simple GUIs
Thank you for listening