Supporting Large-Scale Data-Intensive Computation for Seismology in VERCE

Iraklis Klampanos, Alessandro Spinuso, Luca Trani, Amy Krause, Visakh Muraleedharan, Celine Hadziioannou, Gaia Soldati, Licia Faenza, Lucia Zaccarelli, Peter Danecek, Xavier Briand, Alberto Michelini, Malcolm Atkinson and Jean-Pierre Vilotte

iraklis.klampanos@ed.ac.uk

http://www.verce.eu

Overall Objectives

- VERCE: **V**irtual **E**arthquake and Seismology **R**esearch **C**ommunity in **E**urope
- Unified, Europe-wide computing infrastructure
- Support common ***data***- and CPU-intensive tasks
- Exploit local and remote processing and storage resources
- Use-case driven:
  - Ambient noise cross-correlation

Data-Intensive Requirements

- "Data-intensive": when data management and processing is the bottleneck due to volume or algorithmic reasons.
- Data staging, pre-processing and cross-correlation
  - mainly on private or (in the future) high-throughput resources
- Arbitrarily many continuous waveforms (typically 24h records from local, regional, global networks)
- Over arbitrarily long periods of time
- Unified view of data sources (files, online feeds)
  - Through metadata (e.g. station, channel, time, etc.)

VERCE Technical Goals

- Transparent parallelism and maximum utilisation of local resources
  - Fine-grained *Dispel* workflows
- Data management
  - Preemptive and reactive movement/replication
  - Extensive use of science-specific metadata
- Customisation of workflows
  - Use of community oriented library tools, e.g. ObsPy

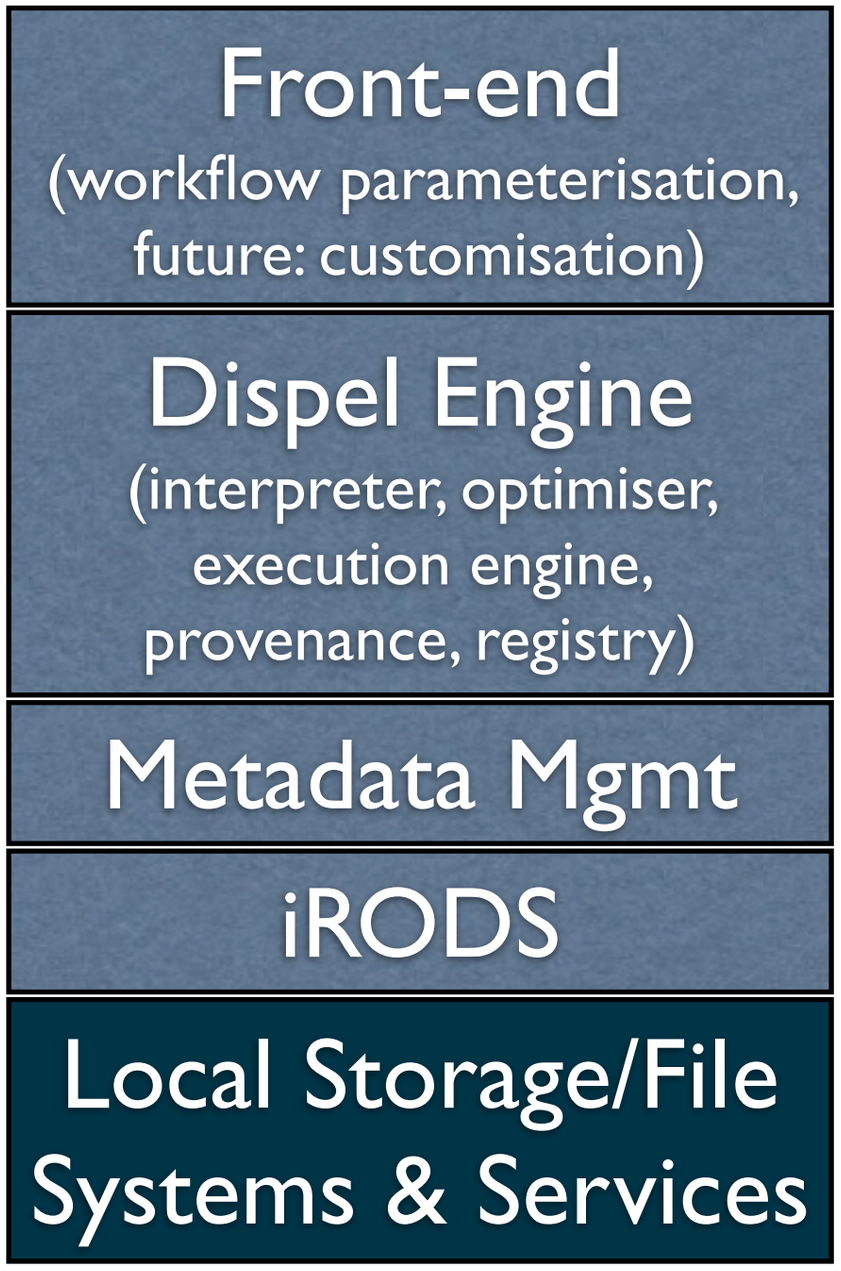
Dispel: Fine-Grained Scientific Workflows

- Composition of arbitrarily fine-grained workflows through *processing elements* and *functions/constructors*
- Stream-oriented / online processing:
  - Keeping 'threads' of workflows busy
  - Decoupling from lower-level, technical details
- An example seismology processing element may have
  - High-level specification in Dispel - typed I/O streams and parameters
  - Lower-level implementation in ObsPy, or other libraries
  - Tweaking parameters alters behaviour

VERCE Data Management

- Main components, currently under consideration:
  - iRODS network in participating sites
  - MonetDB-based metadata manager
- iRODS offers support for decentralised policies
  - different scientific sites have different requirements
- MonetDB component
  - indexes scientific metadata available (through iRODS) at each site
  - unifies data access to VERCE components (e.g. Dispel engines)

Architectural Stack



- D-I workflows are defined in terms of stream-based PEs
  - Executed initially on Dispel-aware, local resources
- Scientific codes and scripts provide the basis for Dispel PEs
  - Forming a shared, distributed library of components
- Data are queried and retrieved, in principle, via scientific metadata

Current Actions

- Evaluation of individual components
  - iRODS, MonetDB, Dispel execution engines
- Enrichment and refinement of the PEs library to meet scientific requirements
  - Currently Python-based, ObsPy
- Collaborating with other projects and initiatives
  - SCI-BUS: support for meta-workflows, potential support for SHIWA for VERCE, etc.
  - EUDAT: archiving of products
  - ERC Whisper for additional data-intensive seismic noise correlation methods and tools

Roadmap and Conclusions

- 6 Months - Alpha testing large-scale cross-correlation
  - VERCE data layer and integration with Dispel engine
  - Basic interaction through a Science Gateway
  - Continuous deployment on VERCE Dispel-aware sites
- 12 Months - VERCE test platform open to public
  - Better integration with archiving services (e.g. EUDAT)
  - High-throughput computing support (EGI), alternative execution engines and optimisation
  - Workflow parameterisation through the front-end
- Combination of 'in-house' development and reaching out
  - maximising impact and sustainability