DISPEL Enactment

Amy Krause¹ and Chee Sun Liew²

¹ EPCC, University of Edinburgh
² Data-Intensive Research Group, University of Edinburgh
DISPEL life-cycle

1. DISPEL development
2. DISPEL language processing
3. Optimisation
4. Deployment
5. Execution & Control
Enactment Model

1. DISPEL language processing
   1. Validation & import from registry
   2. Format & meaning mis-match handling
   3. Interpretation to generate graph

2. Graph optimisation & mapping
   1. Re-ordering & parallelisation
   2. Identification of where to do the work
   3. Selection of PE implementations & instances
   4. Partitioning into co-located subgraphs

3. Deployment & initialisation

4. Execution, Monitoring & Clean up

Preparation to do the data processing

Doing the data processing
Data-Intensive platform

- an application development environment
  - (including libraries of processing elements, functions, and data types),
- a gateway
  - as the entry point of enactment which accepts DISPEL request,
- a DISPEL language processor
  - that compiles the DISPEL request into graph representation,
- an enactment engine
  - that optimises those graphs, deploys them, executes them in a controllable framework that permits interaction with the end user, and finally terminates them and cleans up the environment,
- execution engines
  - that deploy and execute workflows, and
- data sources
  - that are connected and made available through this platform.
Execution model of a DISPEL request

1: Submit DISPEL request

2: Parse request

3: Parse

4: Return graph

5: Optimise and Map graph

6: Transform & Map

7: Return optimised graph

8: Enact graph

9: Create Executable workflows

10: Deploy workflow

11: Initialise PEI

12: Deployment completed

13: Execute workflow

14: Execute PEI

15: Execution completed

16: Enactment Completed

17: Return Results

Metrics for workflow level

Metrics for PEI level
DISPEL enactment process

1. DISPEL Language Processing
   - Parse DISPEL
     - DISPEL
     - Function Execution
       - Graph
     - Type Validation & Annotation
       - Annotated Graph
     - Select PEs & Resource Allocation
       - Annotated Graph w/ Resources

2. Optimisation
   - DISPEL
   - Terminated & Cleanup
     - Results
     - Enact Graph
       - Deployed Graph
     - Insert Monitoring & Deploy Graph
       - Performance Data
         - Performance Database
           - Candidate PEs
             - Performance Data
               - Registry
                 - Function Descriptions
                   - Repository
                     - Function Implementations

4. Execution & Control
   - Termination & Cleanup
     - Results
     - Enact Graph
       - Deployed Graph
     - Insert Monitoring & Deploy Graph
       - Performance Data
         - Performance Database
           - Candidate PEs
             - Performance Data
               - Registry
                 - Function Descriptions
                   - Repository
                     - Function Implementations

DISPEL Enactment - Liverpool, 3 September 2012
Four stages in the enactment process

• DISPEL Language Processing
  – parsing and validating a DISPEL program and creating the data-flow graphs.

• Optimisation
  – selection of PEs, transformation of the data flow graph, substitution of PEs, identification of available resources, and the mapping of PEs to resources.

• Deployment
  – translation into platform-specific form and initialising resources and connections.

• Execution and Control
  – instrumentation and performance measurement, failure management, delivering results and clean up.
Stage 1: DISPEL Language Processing

• Compilation
  – DISPEL request parsing and syntax validation
  – Import used types and objects from the registry

• DISPEL graph generation
  – Interpret the DISPEL request
  – Execute functions, statements, conditionals, loops
  – Connect processing elements to form graphs

• Registration
  – Types and objects may be registered in the ADMIRE registry for re-use
Stage 2: Optimisation

- Graph transformation
  - Sub-graph substitution
  - Parallelisation
  - Reordering
- Resource allocation - where to do the work
  - Mapping data resources to physical resources
  - Partitioning the graph
  - Assigning sub-graphs to worker nodes
Stage 2: Optimisation

• Gateway executes a chain of optimisers
• Extension point for developers to plug in their own implementations
  – Handle specific cases
  – Example: Chee Sun’s measurement-based optimiser for resource allocation
Stage 3: Deployment

• Creating concrete (executable) workflows
  – Mapping PEs to concrete processing units
  – Linking data connections between PEs
  – Configuring data transfers between execution engines (data marshalling)

• Distributed processing
  – Single vs. multiple execution engines
  – Single vs. multiple gateway
Stage 4: Execution and Control

• Processing control
  – Distribute sub-processes to worker nodes
  – Monitor activity of distributed processes

• Data-flow control
  – Initiate and manage data transfer between nodes
  – Publish data sources for result retrieval

• Termination, error management and cleanup
  – Collect and publish progress reports and errors
  – Best effort for cleaning up resources