Distributed Programming Abstractions

Shantenu Jha

M Cole, D Katz, M Parashar, O Rana & J Weissman

http://wiki.esi.ac.uk/Distributed_Programming_Abstractions

http://www.cct.lsu.edu/~sjha/dpa_publications/
Overview

- **DPA Intellectual Advances**
  - Survey the Landscape of Distributed Application (DA) and CI
  - Introduce IDEAS as minimal set of DA Design Objectives

- **DPA in Practice**
  - IDEAS Influence development of DA through SAGA
  - Pilot-Job: Power of “right” Abstractions

- **DPA Retrospective**
  - DPA: Why a Research Theme?
  - Events, Workshops and Publications
  - Beyond DPA: 3DPAS
DPA: Scope

- DPA is concerned about making it easier to develop scientific DA
  - How they are developed? Is there a theoretical framework?
  - Role of Distributed Computing Infrastructure?
    - “Programming” the infrastructure (services, middleware..)
  - Understand role of patterns and abstractions
    - “Programming” an application one aspect scientific DA
  - End-to-end: Developing, Deploying and Execution

- DPA Application Scope
  - Resource Intensive – require multiple/concurrent resources
  - Study of representative set of applications
    - S&E on DCI, not bespoke infrastructure; single User Mode
  - Distinct from “simple” HPC applications
    - Not just peak-performance, varying control over exec env
  - Applications do not typically have: Security, Stringent QoS
DPA: Methodology

- DPA Intellectual and analytical framework to understand the design, development & deployment of DAs within the context of production CI

- Construct the Landscape of e-Science Applications and Infrastructure
  - Studied / Investigated 24-30 Applications
  - Find commonalities: Identify set of independent characteristics
  - Refine set of applications (group) using basis-set/vectors; refine basis-set/vectors using better understanding of applications
  - Gap Analysis:
    - Analysis programming systems, tools and Infrastructure
    - Analyse development, deployment and execution stages

- Possible Solutions:
  - Ex 1: Support for Usage Modes ("execution patterns")
  - Ex 2: Support for X number of jobs, with given wait-bounds (SLA)
DPA: Primary Observations

- Is large (and rich), but the number of effective and extensible DA small
  - More than just submitting jobs here and there!

- Developing DA is a hard undertaking
  - Intrinsic and Extrinsic Factors
  - Unique role of the Execution Environment (Infrastructure)

- Embrace “distributedness”
  - Understanding distributedness, heterogeneity & dynamic execution is fundamental (e.g., Exascale logically distributed prog. Models)
  - Data-centric application will be the drivers!

- Role for Pattern-oriented and Abstractions-based Development

- Autonomic approaches required to manage complexity and dynamism
Assertion #1: The space of possible DA is large, but number of effective DA small

- Distributed Application: That need multiple resources, or can benefit from the use of multiple resources;
  - can benefit from increased peak performance, throughput, reduced time-to-solution
  - More than just HPC or HTC Applications
    - e.g., DDDAS scenarios

- Ability to develop simple or effective distributed applications is limited
  - Applications that utilize multiple resources sequentially, concurrently or asynchronously is low

- Developing DA > just submitting jobs to remote sites!
  - What the pieces of distribution are? How these pieces interact? Flow of information? What is needed to actually deploy and execute the application?
Assertion #2: Developing DA is a hard undertaking

- **Intrinsic reasons why developing DA is fundamentally hard:**
  - Control & Coordination over Multiple & Distributed sites
    - Effective coordination in order for whole > sum of the parts
  - Complex design points; wide-range of models of DA
    - Many reasons for using DA, more than (just) peak performance

- **Extrinsic:**
  - Execution environments will be dynamic, heterogeneous and varying degrees-of-control
    - Fundamental different variation in role of Execution Environment–distinguishing feature of DA from “regular environment” HPC
  - Application types strongly coupled to the infrastructure capabilities, abstractions/tools, & policy:
    - Often development tools assume “specific” deployment and execution environments, or don’t where needed!
    - Policies and tools, e.g production DCI has been missing for DDDAS
Assertion #2: Developing DA is a hard undertaking

- Distributed Application (DA) Vectors:
  - What the pieces of distribution are? How these pieces interact? Flow of information? What is needed to actually deploy and execute the application?

- DA Vectors: Preliminary theoretical framework to analyze DA structure

- Vectors: Axes representing application characteristics; understanding the value helps:
  - App Requirements
  - Skillful aggregation versus Decomposition
  - Primacy of coordination
  - Design, constraints of solutions

<table>
<thead>
<tr>
<th>Application Example</th>
<th>Execution Unit</th>
<th>Communication (Data Exchange)</th>
<th>Coordination</th>
<th>Execution Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Montage</td>
<td>Multiple sequential and parallel executables</td>
<td>Files</td>
<td>Dataflow (DAG)</td>
<td>Dynamic process creation, workflow execution, file transfer</td>
</tr>
<tr>
<td>NEKTAR</td>
<td>Multiple concurrent instances of single executable</td>
<td>Messages</td>
<td>SPMD</td>
<td>MPI, co-scheduling</td>
</tr>
<tr>
<td>Coupled Fusion Simulation</td>
<td>Multiple concurrent parallel executables</td>
<td>Stream-based</td>
<td>Dataflow</td>
<td>Co-scheduling, data streaming, async. data I/O</td>
</tr>
<tr>
<td>Asynchronous Replica-Exchange</td>
<td>Multiple sequential and/or parallel executables</td>
<td>Pub/sub</td>
<td>Dataflow and events</td>
<td>Decoupled coordination and messaging, dynamic task generation</td>
</tr>
<tr>
<td>Climate-Prediction.net (generation)</td>
<td>Multiple sequential executables, distributed data stores</td>
<td>Files and messages</td>
<td>Master/worker, events</td>
<td>AtHome (BOINC)</td>
</tr>
<tr>
<td>Climate-Prediction.net (analysis)</td>
<td>A sequential executable, multiple sequential or parallel executables</td>
<td>Files and messages</td>
<td>Dataflow (Forest)</td>
<td>Dynamic process creation, workflow execution</td>
</tr>
<tr>
<td>SCOOP</td>
<td>Multiple different parallel executables</td>
<td>Files and messages</td>
<td>Dataflow</td>
<td>Preemptive scheduling, reservations</td>
</tr>
</tbody>
</table>
Assertion #2: Developing DA is a hard undertaking

- Large number programming systems, tools and environments
  - Lack of extensible functionality, interfaces & abstractions
    - Interoperability and extensibility become difficult
    - Art of tool building needs to be more of science!

- Applications have been brittle and not extensible:
  - Tied to specific tools and/or programming system
  - Large number of Incomplete Solutions!

- Unique Role for abstractions for DA and CI
  - Application formulation, development and execution must be less dependent on infrastructure & provisioning details
    - Abstractions for Development, Deployment & Execution
  - A Pattern-Oriented, Abstractions-Based Approach
    - “Abstractions allows innovation at more interesting layers”
Assertion #3: Embrace Distribution

- “History of computing like pendulum, swings from centralized to distributed”
  - Indications this time there is a fundamental paradigm shift due to DATA
  - Too much to move around; learn how to do analytics/compute in situ

- Decoupling and delocalization of the producers-consumers of computation
  - Localized special services; people and collaborations are distributed

- (Ironically) Most applications have been developed to hide from heterogeneity and dynamism; not embrace them
  - Programming models that provide dynamic execution (opposed to static), address heterogeneity etc
  - Logically vs Physically Distributed: NG programming models will need to support dynamic execution, heterogeneity at a logically-distributed level
Assertion #3: Embrace Distributedness
Corollary: Clouds are not Panacea

- Clouds: Novel or more of the same?
  - Better control over software environment via virtualization
  - Illusion of unlimited and immediate available resource can lead to better capacity planning and scheduling
    - Partly due to underlying economic model and SLAs

- Clouds do not remove many/all of the challenges inherent in DA
  - Clouds are about provisioning, grids are about federation
  - Fundamental challenges in distribution remain
    - Makes some thing worse as impose a model of strong localization
    - “The reason why we are so well prepared to handle the multi-core era, is because we took the trouble to understand and learn parallel programming” – Ken Kennedy

- Clouds part of a larger distributed CI
  - Certain tasks better suited for Grids, others on Clouds
Assertion #4: Role for a Pattern-Oriented and Abstraction-Based Development Cycle

- Relation between Application, Abstractions and Patterns:
  - Application: Need or can use >1 R
  - Patterns: Formalizations of commonly occurring modes of computation, composition, and/or resource usage
  - Devel, Deploy & Exec Phase
  - Abstractions: Process, mechanism or infrastructure to support a commonly occurring usage

<table>
<thead>
<tr>
<th>Coordination</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master-Worker (TF, BoT)</td>
<td>Replication</td>
</tr>
<tr>
<td>All-Pairs</td>
<td>Co-allocation</td>
</tr>
<tr>
<td>Data Processing Pipeline</td>
<td>Consensus</td>
</tr>
<tr>
<td>MapReduce</td>
<td>Brokering</td>
</tr>
<tr>
<td>AtHome</td>
<td></td>
</tr>
<tr>
<td>Pub-Sub</td>
<td></td>
</tr>
<tr>
<td>Stream</td>
<td></td>
</tr>
</tbody>
</table>
Assertion #4: Role for a Pattern-Oriented and Abstraction-Based Development Cycle

- Analysis of Distributed Applications leads to three types of patterns
  - Patterns that appear in the Parallel Programming
  - Patterns driven by distributed concerns (e.g., @HOME, consensus)
  - Patterns addressing distributed environment concerns exclusively (e.g., co-allocation)

- There exist tools that support patterns, i.e., provide abstractions

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Tools That Support the Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master/Worker-TaskFarm</td>
<td>Anka, Nimrod, Condor, Symphony, SCE, HPCS</td>
</tr>
<tr>
<td>Master/Worker-BagOfTasks</td>
<td>Comet-G, TaskSpace, Condor, TSpaces</td>
</tr>
<tr>
<td>All-Pairs</td>
<td>All-Pairs</td>
</tr>
<tr>
<td>Data Processing Pipeline</td>
<td>Pegasus/DAGMan</td>
</tr>
<tr>
<td>MapReduce</td>
<td>Hadoop, Twister, Pydoop</td>
</tr>
<tr>
<td>AtHome</td>
<td>BOINC</td>
</tr>
<tr>
<td>Pub-Sub</td>
<td>Flaps, Meteor, Narada, Gryphon, Sienna</td>
</tr>
<tr>
<td>Stream</td>
<td>DART, DataTurbine</td>
</tr>
<tr>
<td>Replication</td>
<td>Giggle, Storm, BitDew, BOINC</td>
</tr>
<tr>
<td>Co-allocation</td>
<td>HARC, GUR</td>
</tr>
<tr>
<td>Consensus</td>
<td>BOINC, Chubby, ZooKeeper</td>
</tr>
<tr>
<td>Brokers</td>
<td>GridBus, Condor matchmaker</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Example</th>
<th>Coordination</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Montage</td>
<td>TaskFarm, Data Processing Pipeline</td>
<td>-</td>
</tr>
<tr>
<td>NEKTAR</td>
<td>-</td>
<td>Co-allocation</td>
</tr>
<tr>
<td>Coupled Fusion Simulation</td>
<td>Stream</td>
<td>Co-allocation</td>
</tr>
<tr>
<td>Async RE</td>
<td>Pub/Sub</td>
<td>Replication</td>
</tr>
<tr>
<td>Climate-Prediction (generation)</td>
<td>Master/Worker, AtHome</td>
<td>Consensus</td>
</tr>
<tr>
<td>Climate-Prediction (analysis)</td>
<td>MapReduce</td>
<td>-</td>
</tr>
<tr>
<td>SCOOP</td>
<td>Master/Worker, Data Processing Pipeline</td>
<td>-</td>
</tr>
</tbody>
</table>
Assertion #5: Need an Autonomic Approach to managing complexity and dynamism

- Challenges in Distributed Applications:
  - **Complexity**: Multiple-levels, External Dependency, Varying control
  - **Dynamism**: Changing resource availability and requirements

- Posit that Autonomic Approaches can help address two challenges
  - Provide formulations that hide complexity & support dynamism
    - Programming the Infrastructure “autonomically”

- Objective: Intelligence in Compute-Data placement
  - Sophisticated models of data-compute co-movement
    - Tier 0 data at BNL; all compute migrated to Tier 0

- For an objective, which strategy?
  - **S1**: Assignment of workers determined by min \( T_r \)
  - **S2**: Assignment of workers, by min data transfer
  - **S3**: Upon tracked dependencies

- Each SN: Could have different mechanism
IDEAS: DA Development Objectives

- **Interoperable**: Ability to work across multiple resources concurrently
  - Includes jobs submission, coordination mechanism,

- **Dynamic**: Beyond legacy static execution & resource allocation models
  - Decisions at both deployment and run-time
  - Dynamical execution is almost fundamental at scale

- **Extensible**: Support new functionality & infrastructure without wholesale refactoring, i.e., lower coupling to tools & infrastructure

- **Adaptive/Autonomic**: Flexible response to fluctuations in dynamic resources, availability of dynamic data

- **Scalable**: Along many dimensions and design points

Challenge: To develop DA effectively and efficiently with IDEAS as first class objectives with simplicity an over-aching concern
DPA Theme: Overview

- **DPA Intellectual Advances**
  - Survey the Landscape of Distributed Application and CI
  - Introduce IDEAS as minimal set of DA Design Objectives

- **DPA In Practice**:  
  - IDEAS Influence development of DA through SAGA
  - Pilot-Job: Power of “right” Abstractions

- **DPA Retrospective**  
  - DPA - Why a Research Theme?
  - Events, Workshops and Publications
  - Beyond DPA - 3DPAS
There exists a lack of Programmatic approaches that:

- Provide general-purpose, basic & common grid functionality for applications; hide underlying complexity, varying semantics.
- The building blocks upon which to construct “consistent” higher-levels of functionality and abstractions
- Meets the need for a Broad Spectrum of Application:
  - Simple scripts, Gateways, Smart Applications and Production Grade Tooling, Workflow...

Simple, integrated, stable, uniform and high-level interface

- Simple and Stable: 80:20 restricted scope and Standard
- Integrated: Similar semantics & style across
- Uniform: Same interface for different distributed systems

SAGA: Provides Application* developers with units required to compose high-level functionality across (distinct) distributed systems

(*) One Person’s Application is another Person’s Tool
IDEAS: Developing Distributed Application From First Principles?

- How to develop a simple MR that is interoperable across infrastructure concurrently?
  - First Principles Programming
  - What abstractions are provided
    - Interoperable M-W Pattern?

- Understand the determinants of performance and the trade-offs
  - Test assumptions & check approximations
  - Reason about an application

- Same application, different programming models
  - Very different performance dependence
IDEAS: Replica-Exchange Simulations

- Replica-Exchange (RE) methods:
  - Represent a class of algorithms that involve a large number of loosely coupled ensembles.

- RE simulations are used to understand a range of physical phenomena:
  - Protein folding, unfolding etc
  - MC simulations

- Many successful implementations:
  - Eg folding@home [replica based]
IDEAS: Abstractions for Dynamic Execution: Container Task
Abstractions for Dynamic Execution (2)
SAGA Pilot-Job (BigJob)
Distributed Adaptive Replica Exchange (DARE) Multiple Pilot-Jobs on the “Distributed” TeraGrid

- Ability to dynamically add HPC resources. On TG:
  - Each Pilot-Job 64px
  - Each NAMD 16px

- Time-to-completion improves
  - No loss of efficiency

- Time-per-generation is measure of sampling

- Variants of RE: Sync (local) vs async (distr.)

Luckow, Kim, Schnor, Jha
Adaptive Replica-Exchange, Phil. Trans of Royal Society A (2009)
Deployment & Scheduling of Multiple Infrastructure Independent Pilot-Jobs
IDEAS: Facilitating Novel Execution Modes

- Interoperability and Scale-out enable new ways of resource planning and application execution

- Deadline-driven scheduling: e.g., task done before time $T$

- Adapt workload distribution and resource utilization to ensure completion

**Table I: Usage of Cloud Pilot-Jobs to Ensure Deadline**

<table>
<thead>
<tr>
<th>Result</th>
<th>#Occurrences</th>
<th>Average $T_G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>No VM started</td>
<td>6</td>
<td>7.8 min</td>
</tr>
<tr>
<td>1 VMs started</td>
<td>1</td>
<td>36.4 min</td>
</tr>
<tr>
<td>2 VMs started</td>
<td>1</td>
<td>47.3 min</td>
</tr>
<tr>
<td>3 VMs started</td>
<td>2</td>
<td>44.2 min</td>
</tr>
</tbody>
</table>
Azure: Using BigJob API and Coordinating Multiple Tasks

- For Replica-Exchange application, take SAGA-BigJob and implement on Azure to test for performance

- CONCLUSION: For same workload, comparable in performance to TG!
  - BUT SIMPLER to implement than TG!
Three types of storage abstractions:
- **Azure Blob Service**: Large amounts of raw data
  - Block Blob: Large chunks of 5GB
  - Page Blob: Manage storage as an array
- **Azure Table Storage**: Semi-structured data
- **Azure Queue Storage**: Message Queues
- All three replicated and with strong consistency semantics

Current affinity operates at the data-center level;
- Enhanced/finer-grained affinity to be made available
Corollary to Assertion #3 Revisited: Clouds are not Panacea

- Nascent infrastructure do what Production DCI have not managed!

- “We” have lost intellectual leadership to Commercial providers!
  - Community busy agenda pushing, e.g., middleware/software stacks, most innovation in DC has come from commercial sector!
  - Commercial Providers seem to have appreciated the basic elements of distributed environments and applications

- Simplicity is the Ultimate Sophistication
  - “It seems that perfection is attained not when there remains nothing to add, but when there remains nothing to remove”
  - Confuse Functionality with Usability
    - “Confused Beauty for Truth”, -- “How Did Economists Get It So Wrong?”, in analogy to the Financial Crisis

- Moral and Intellectual courage to admit we got it wrong
  - More critical analysis in Grid2009, “Critical Analysis of PGI and DA”
IDEAS: Framework for Dynamic Execution of Multi-Physics (FeDEX)

- Pilot-Job: execution environment for multi-physics coupled simulations
  - Avoids multiple waiting times & co-scheduler (TG 5+ yr: NO)
  - Simple Dynamic Resource Assignment as simulation progress
FeDEX for Multi-physics Simulations Logical vs Physically Distributed

• S1-Local (Left): CFD-MD start within 1 BJ. As 2nd BJ becomes available on same resource, CFD-MD execution trivially migrates to 2 BJ (CFD-MD in diff. BJs).

• S2-Dist (Right): CFD-MD start within 1 BJ. As 2nd BJ starts on distributed resource, FEDEX distributes CFD-MD execution

• No reprogramming/difference for Application User
What is unique about Pilot-Jobs built using the right abstractions?

- **Pilot-Jobs:** Decouple Resource Allocation from Resource-Workload
  - Enhancing resource utilisation
  - Lowering wait time for multiple jobs (better predictibility)
  - Facilitate high-throughput simulations

- Two unique aspects about the SAGA-based Pilot-Job:
  - **Pilot-Jobs have not been used for Science Driven Objectives:**
    - First demonstration of enhanced sampling
    - First demonstration of supporting multi-physics simulations
  - **Infrastructure Independent**
    - Falkon, Condor Glide-in, Ganga-Diane, DIRAC/WMS, PANDA
    - Frameworks based upon pull-model/PJs for specific back-end
    - Not all support MPI

- SAGA-based Pilot-Job form the basis:
  - For autonomic scheduling and resource selection decisions
  - RT frameworks for load-balancing and fault-tolerance
IDEAS: CI to Support DDDAS
EnKF -- Model Inversion

1. Parameter Sampler
2. Task Farming
3. Flow Model Construction
4. Task Farming
5. Evolve to Assimilation
6. Compute Kalman Gain
7. Task Farming
8. Apply Gain to States
9. Update Ensemble Models

Flow Model Parameters
Math Model Metadata
New Sensor Data?
Yes
No
Observation Data from Sensors
Sensor Metadata
Covariance Analysis
Observation Error Model
Ensemble Metadata
Yes
No
Kalman Gain
All Converged?
BQP allows users to make bound predictions of how much time a job of a given size, duration and location will spend in the queue.

The prediction is given with a degree of confidence (probability) and quantile (repeatability).

BQP is available on a number of TeraGrid resources.

SAGA-based framework uses external service -- BQP information, to submit BigJobs with sizes, durations to locations that are optimized to spend a minimal amount of time in the queue, therefore a smaller total time to completion.
Results: Scale-Out Performance

- Using more machines decreases the TTC and variation between experiments.
- Using BQP decreases the TTC & variation between experiments further.
- Lowest time to completion achieved when using BQP and all available resources.

- Khamra & Jha, GMAC, ICAC’09
- TeraGrid Performance Challenge Award 2008
Performance Advantage
Why Autonomics?

Forward Vs Reverse Scheduling
- **Forward**: Take a workload; find a set of resources
- **Reverse**: Find a set of resources (first-to-run) to within constraint; tune workload
- Or match-the workload (adaptive)
- **Dynamic Execution is underlying paradigm**
  - Resource selection and late binding to resource
  - Optimal resource configuration not just resource selection

---

<table>
<thead>
<tr>
<th>Sample #</th>
<th>Machine</th>
<th>Queue</th>
<th>Num. Cores</th>
<th>Duration (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>Ranger</td>
<td>normal</td>
<td>64</td>
<td>2:00</td>
</tr>
<tr>
<td>1-10</td>
<td>Ranger-BQP</td>
<td>development</td>
<td>128</td>
<td>1:00</td>
</tr>
</tbody>
</table>

*Table 1*: Table showing the configurations chosen on Ranger, with and without BQP. Notice how the use of BQP has a small, but significant change in the queue, size and duration requested.

<table>
<thead>
<tr>
<th># of Samples</th>
<th>Machine</th>
<th>Queue</th>
<th>Num. Cores</th>
<th>Duration (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Ranger</td>
<td>development</td>
<td>64</td>
<td>1:30</td>
</tr>
<tr>
<td>5</td>
<td>Ranger</td>
<td>development</td>
<td>64</td>
<td>2:00</td>
</tr>
<tr>
<td>2</td>
<td>Ranger</td>
<td>development</td>
<td>128</td>
<td>1:00</td>
</tr>
<tr>
<td>3</td>
<td>QueenBee</td>
<td>checkpoint</td>
<td>16</td>
<td>2:00</td>
</tr>
<tr>
<td>4</td>
<td>QueenBee</td>
<td>checkpoint</td>
<td>16</td>
<td>1:00</td>
</tr>
<tr>
<td>3</td>
<td>QueenBee</td>
<td>checkpoint</td>
<td>32</td>
<td>1:00</td>
</tr>
<tr>
<td>3</td>
<td>Abe</td>
<td>dque</td>
<td>64</td>
<td>2:00</td>
</tr>
<tr>
<td>7</td>
<td>Abe</td>
<td>dque</td>
<td>64</td>
<td>1:00</td>
</tr>
</tbody>
</table>

*Table 2*: Table showing the selected configuration of the resources and the number of times a particular configuration is chosen, when the decision is guided by BzaQP. Data in this table corresponds to RQA-BQP; the experiments are repeated ten times. As can be seen, the use of BQP results in a varying choice of resource configuration on different machines. In contrast, when BQP is not used, a fixed configuration is employed.
Adaptivity: Applications & Infrastructure-level

Scaling for 20x20x20

Number of cores (c1.medium, 2 cores per VM)

Time (seconds)

Scaling for 20x20x20

Number of cores (c1.xlarge, 8 cores per VM)

Time (seconds)

Application Adaptivity (stage2)

Time (minute)

Deadline (minute)

Infrastructure Adaptivity (stage2)

Time (minute)

Deadline (minute)
Rep-Ex tied to specific implementations and infrastructure

What are the limits to scalability of Rep-Ex Algorithm?
- Traditionally, infrastructure limitations; also implementation doesn’t support extensions
- Classical conundrum: Hence no attempts at algorithmic innovation, thus no pressure on infrastructure to support!

Can we implement flexible, extensible and scalable RE capability?
- Interoperable: Usage across multiple infrastructure
- Extensible -- to new methods of comm. & coordination
  - Supports different replica pairing/exchanging mechanisms:
    - Synchronous versus asynchronous, variants of Rep-Ex
  - Break the coupling between application & infrastructure:
IDEAS: Rep-Ex Challenges and Opportunities Influencing the Practice of e-Science

- RE Commonly used Method in Biological Simulations
  - ~250+ papers a year by the community
  - Enhanced sampling to faster “un-folding”
  - All implementations are (i) bound to specific infrastructure (ii) fixed coordination mechanism

- Interplay of coordination, communication & infrastructure
  - Asynchronous versus Synchronous
  - Coord. by centralised data-store (polling) versus pub-sub
    - The right choice enables better & faster scientific
  - Distributed versus Localised

- Co-design of formalism, abstractions & dynamic execution
  - Pilot-Jobs and Pilot-Data
  - Test for the IDEAS design objective
Synchronous (Traditional) RE

- For the traditional implementation of RE and number of replicas (N), create a **fixed set of \( \frac{N}{2} \)** pairs of replicas.
  - When *all* the replicas reach a pre-determined state, the exchanges are attempted (the exchange step).
    - If yes, (i.e. exchange is successful), swap temperature & restart
    - If not, continue till next exchange

- Limitations:
  - Exchanges can only take place between fixed paired replicas
  - Inhibits exchanges between replicas with non-nearest temperatures.
  - **Synchronized exchange steps**
    - Inefficient for heterogeneous infrastructure (i.e., different running times for each replica).
  - **Synchronized exchange step means concurrent (bursty) communication and coordination**
    - Does not scale for large N
Implement Asynchronous version of the RE algorithm to overcome the limitations.

- Replicas can perform exchanges with any other available replica
  - Whenever possible – instead of waiting for a synchronized exchange step
  - Any two replicas can attempt to exchange.

Experiment with two different Async implementations:

- Centralized point of decision making & coordination (cent)
  - A master co-ordinates and manages all the replicas and exchanges [Classical M-W Paradigm]

- Decentralized coordination (De-cent)
  - Each replica is handled independently [akin to P2P..w/ M]
  - Does this prevent central manager/master from becoming a bottleneck, with large N?
Scale-Up Performance: Increasing N, 1 Resource
Scale-Out Performance: Performance with increasing number of resources (TG + LONI)
DPA Retrospective

Surveying the Landscape of e-Science at DPA Workshop @ Europar
DPA: Why a theme?

- DPA is about analysing applications effectively and efficiently, with extensibility, scalability & adaptivity as 1st Class concerns

- DPA has provided opportunity to address questions/issues that are not possible as single investigator!
  - Difficult Questions! And not the “usual” Research Papers
  - “Critical Perspectives of Dist. Applications & Production CI”
  - Input requires Multiple Areas of expertise diverse backgrounds

- Multiple Areas of Intellectual Output and contribution:
  - “Abstractions for Distributed Applications and Systems”
  - Contributed (directly and indirectly) broad range of problems
  - Global Recognition of the DPA Brand & Expertise

- Influenced many researchers for many years to come!
  - DPA 3, 5 and 10 year Retrospective? Edinburgh? Banff?
DPA: Events, Workshops & Publications

- Events: Approximately 20
  - (i) Community Engagement Workshops, (ii) Co-location with other Conferences/Meetings, (iii) Focussed Group Meetings
- Approximately 20 Publications, many at the 20% acceptance
- Publications can be categorised into three types:
  - **Type I**: Directly emanating from DPA meetings
    - Book: Wiiley (Dec 31 2010) & Europar 2008 Workshop
    - Grid2009 (Best Paper Award at IEEE Grid2009)
    - Survey Paper, Technical Report, Autonomics Papers
  - **Type II**: Derived & motivated by ideas from DPA
    - Abstractions, Frameworks and novel app. formulation
    - Coordination across Infrastructure (eg ACM ScienceCloud)
  - **Type III**: Indirectly influenced by DPA thinking
    - SAGA developments for Interoperable & scalable implementations of patterns
Beyond DPA

- Lead to an understanding of Autonomic Computational Science
  - Conceptual Framework for Autonomic Applications
  - Defined the concept of Self-* Patterns for Autonomic CS
  - Autonomics Track continues
    - Computing in Science and Engineering [Spring 2011]
    - Scientific Programming, special issue [Spring 2011]
    - ACS Workshop (Brussels) Oct 2010

- “DPA is Dead; Long-live DPA”:
  - Apply research methodology & analysis framework to data-intensive and dynamic applications -- 3DPAS
  - DPA was mostly about “traditional” e-Science: 3DPAS brings autonomics tracks in line with investigation of abstractions for data-intensive research
“Follow-on From DPA”:
- **DPA Theme Methodology**: Survey the Landscape of Distributed Applications, Understand missing abstractions and main challenges of the infrastructure. Generalise to 3D.

3D: Dynamic, Distributed and Data-Intensive
- **Distributed and Data-Intensive Scenarios**
- **Dynamic Data Scenarios: T_D**
  - Sensor-based
  - Location changes: Where to place data? Assume R>1, which data element to pick?
  - T_D < T_A e.g., on-demand, adaptive data

Think Three Tracks -- 3 x 3 Matrix:
- (i) Applications, (ii) Programming Abstractions and Systems (iii) Cyberinfrastructure
- Identify and explore each “D” for each
Embracing the challenge…

… and displaying the “beads of our labour” (DPA workshop at MardiGras Conference 2008)
DPA TEAM: Murray Cole, Daniel Katz, Manish Parashar, Omer Rana and Jon Weissman

Many others who contributed to the early workshops & refining the “big issues”

Malcolm Atkinson for advise and guidance

Elizabeth van der Meer, Anna Kenway, & Staff at e-Science Institute

http://wiki.esi.ac.uk/Distributed_Programming_Abstractions

http://www.cct.lsu.edu/~sjha/dpa_publications/