IBM Systems Research

Computational Steering of HPC Applications on Blue Gene Systems

Bruce D'Amora

IBM T.J. Watson Research Center

KAUST Collaborators:

Ying Qian, Madhu Srinivasan, Dan Lecocq, Iain Georgeson, Bob Danani

Outline

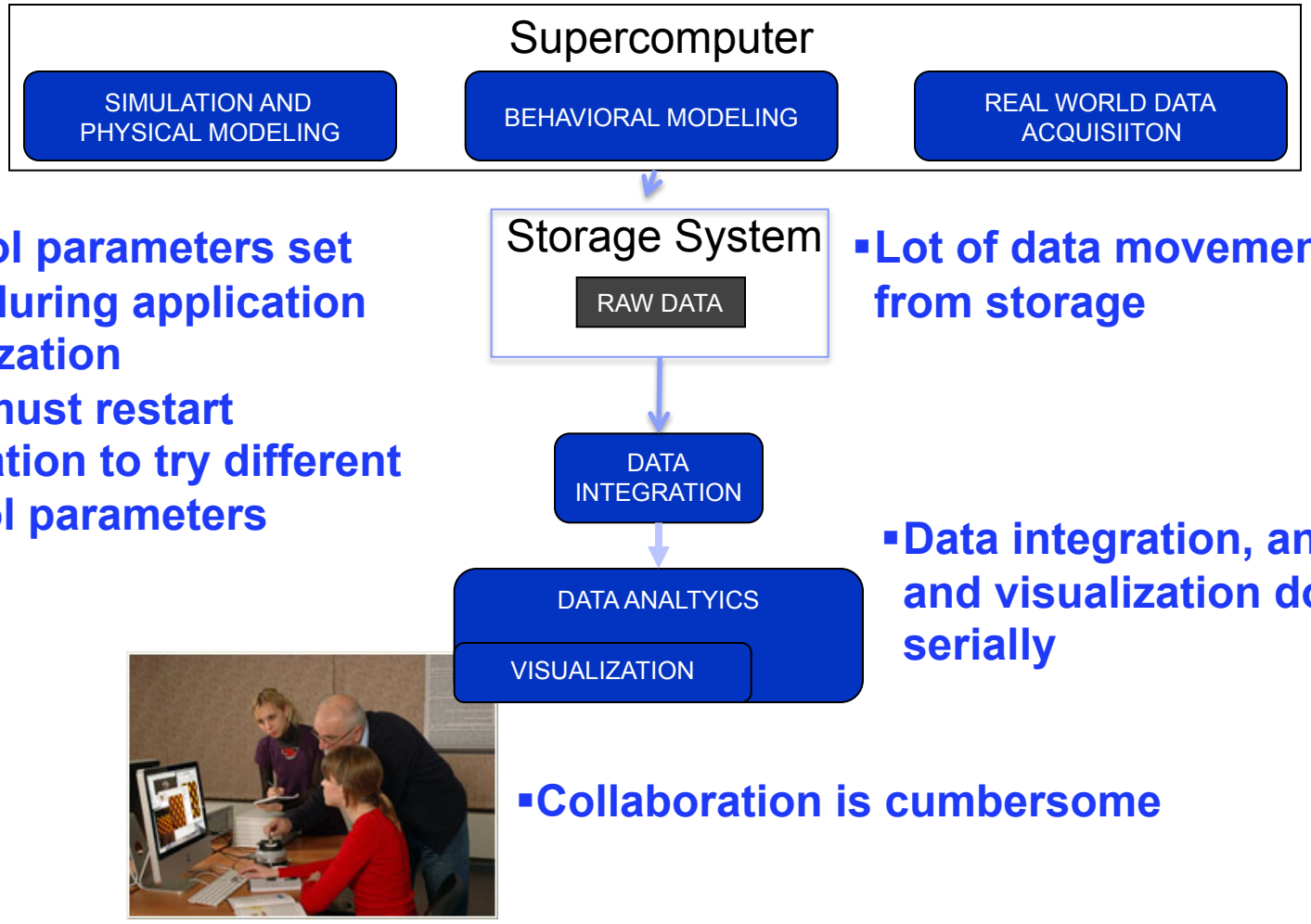
- **Motivation**
- **A Framework**
- **RealityGrid**
- **Blue Gene Implementation**
- **Next Steps**

Motivation for Steering

- **Realtime**
  - Coupling of simulation results with visualization or other downstream apps (analytics)
  - Exploration of results
  - Modification of control parameters

Human Feedback in Simulation Loop

Present Computational Workflow



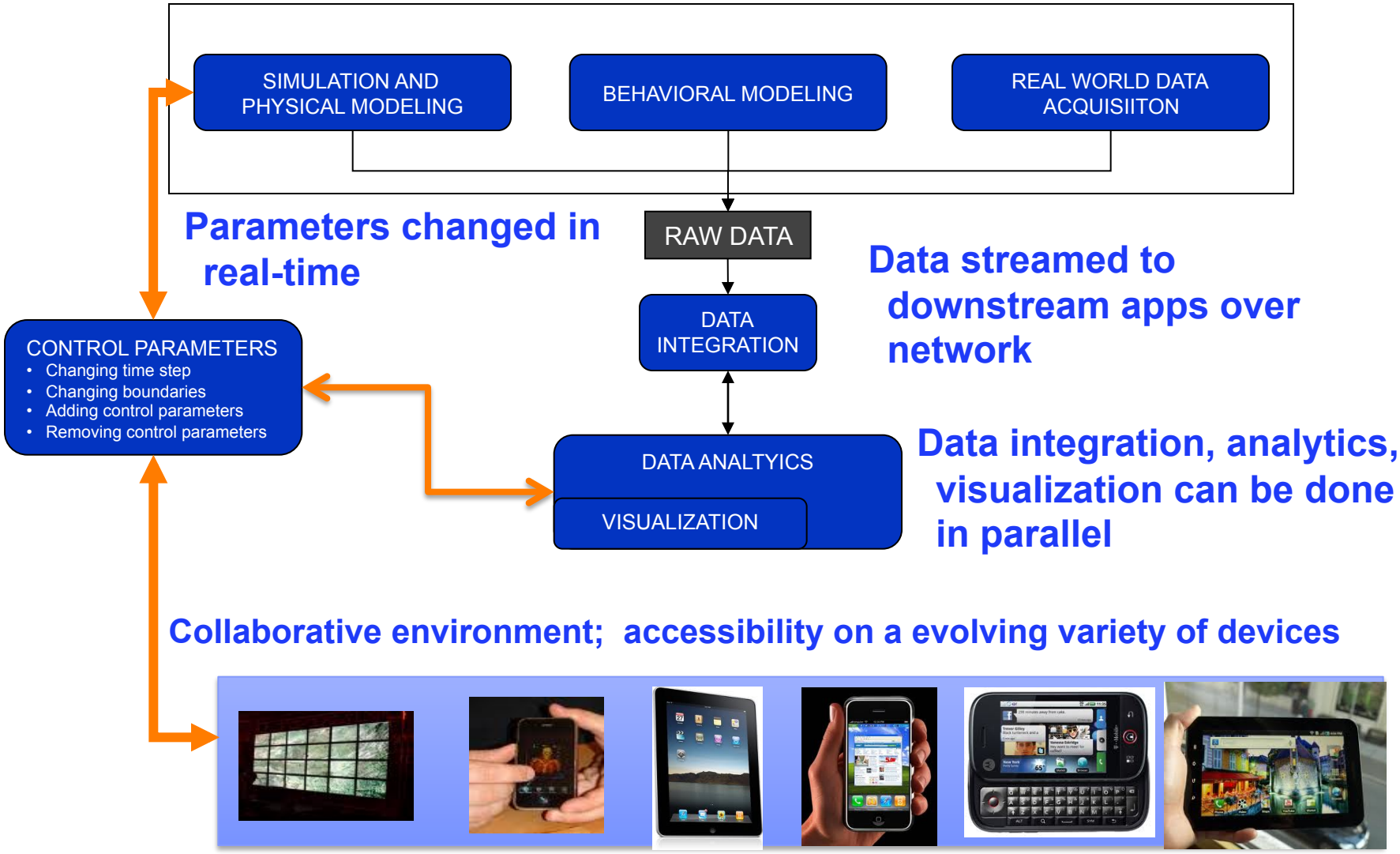
- **Control parameters set once during application initialization**
- **User must restart simulation to try different control parameters**

- **Lot of data movement to/ from storage**

- **Data integration, analytics, and visualization done serially**

- **Collaboration is cumbersome**

Future Computational Workflow



Goals for Steering Framework

Usage Scenarios

- User modifiable control parameters during simulation
- Visualization of intermediate & final results of simulations
- Support collaboration – multiple users at multiple locations

Programmability

- Framework must be "thin"; performance sensitive
- Easy to use; complex APIs delay or completely inhibit adoption
- Easily Extensible – no single API fits every applications needs
- Web services enabled – needs to be accessible in the Cloud
- Open source

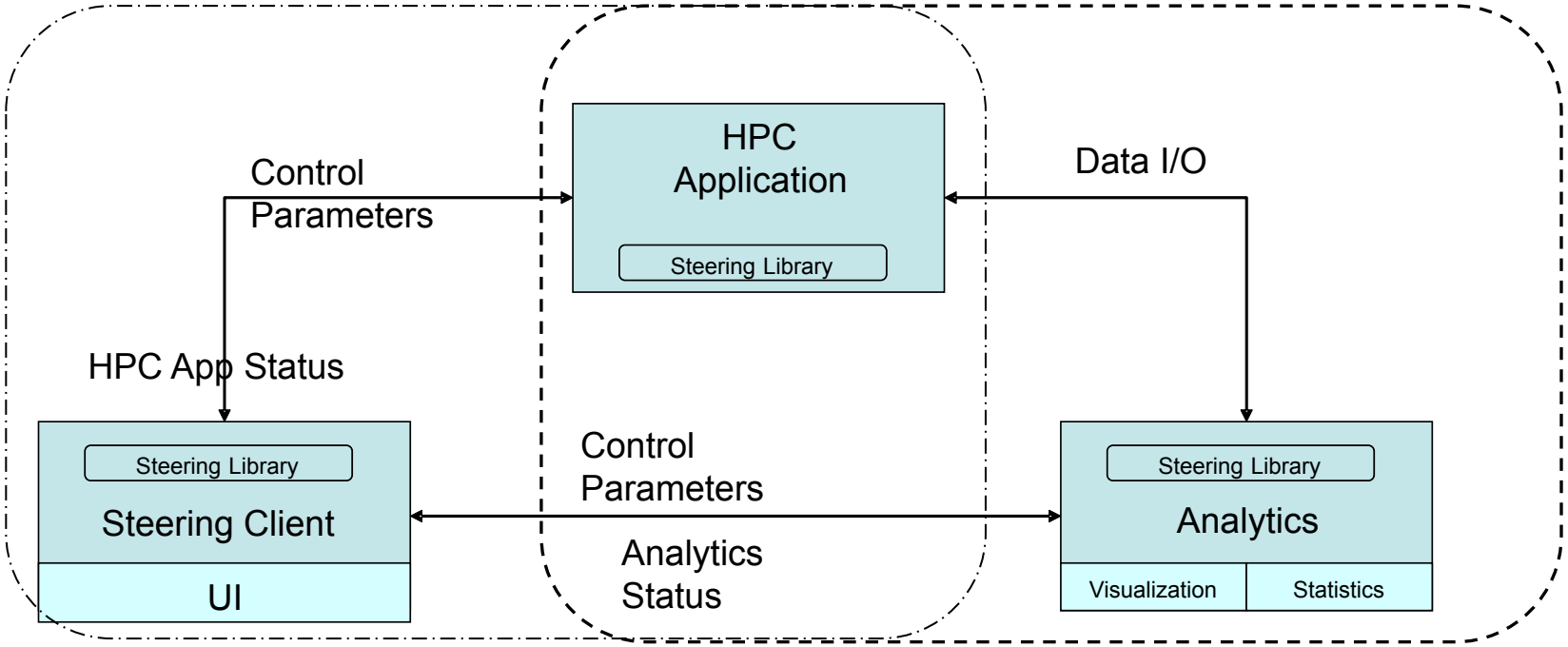
RealityGrid Steering Library

- **Developed with UK Engineering and Physical Sciences Research Council (EPSRC) grant.**
- **Consortium of universities produced BSD licensed Steering Library**
  - University College, London (Centre for Computational Science and Department of Chemistry)
  - University of Edinburgh (Department of Physics and Astronomy and Edinburgh Parallel Computing Centre)
  - University of Manchester (Manchester Computing and Department of Computer Science)
  - Imperial College, University of London (Department of Computing)
  - Loughborough University (Loughborough University Advanced VR Research Centre and Department of Mathematical Sciences)
  - University of Oxford (Department of Materials).
- **The collaborating organizations include the**
  - Computation for Science Consortium (which operates the UK's national supercomputing facility CSAR in Manchester)
  - Schlumberger Cambridge Research Ltd.
  - Edward Jenner Institute for Vaccine Research
  - Silicon Graphics Inc.
  - Advanced Visual Systems Inc.
  - Fujitsu Ltd.
  - BTexact (in collaboration with University of Manchester)
- **Principal Investigator: Peter Coveney, Univeristy College London**

RealityGrid Features

- **Supports 3 types of communications between HPC application and steering clients**
  - Direct socket connection
  - File based communications – closest to current workflow
  - Web Services Resource Framework (WSRF-Lite) interfaces
- **All communications are embedded in xml streams**
- **WSRF provides persistent state and capability of registering multiple HPC applications to the grid framework**
  - Clients can pick which HPC applications they wish to connect with via web interface
- **Includes bindings for C/C++, Fortran, python, perl, java, wrappers**
- **VTK data reader class developed**

RealityGrid Steering Framework

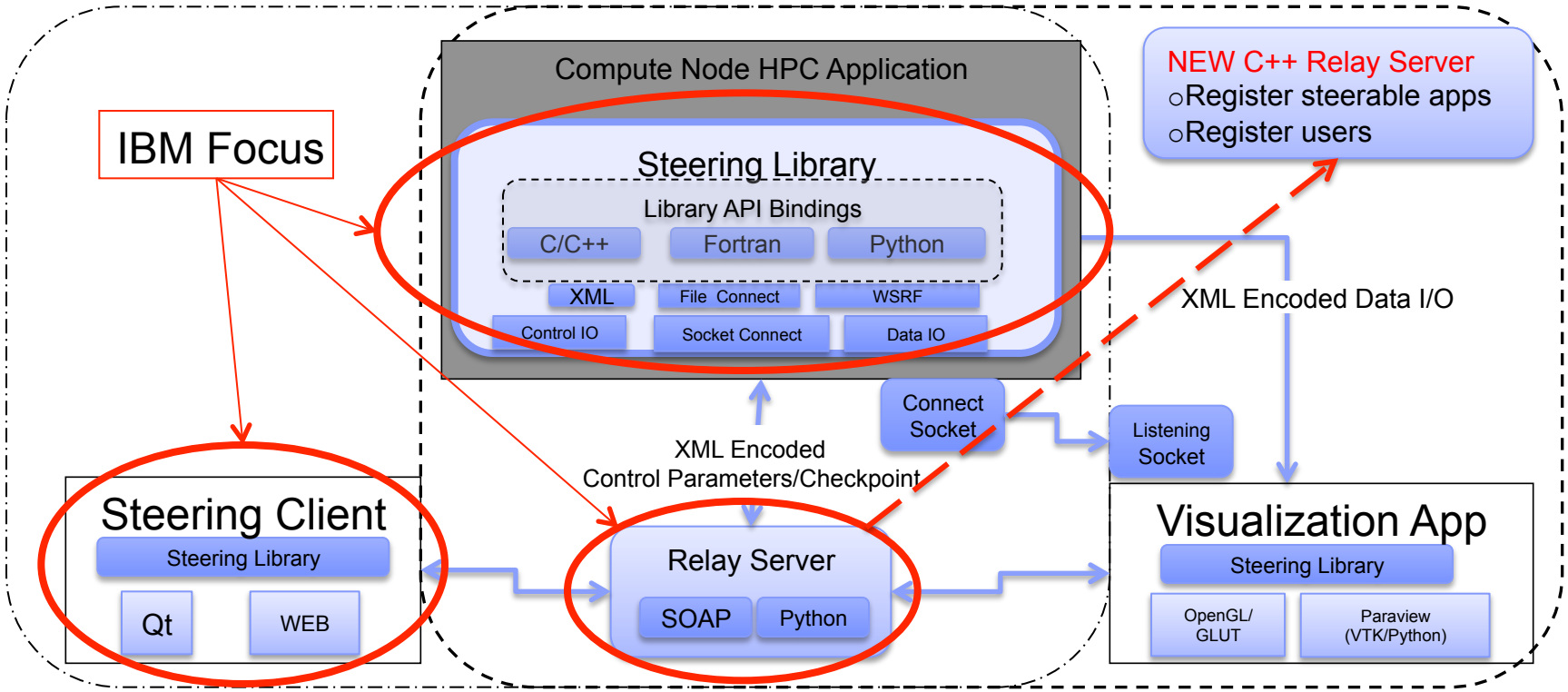


- **Steering Library Function** include, but are not limited to:
  - Identify connections between HPC, Steering, Analytics
  - Identify control parameters as read, write, update
  - Identify data as read, write, update, float, char, integer, etc.
  - Support multiple connections from different clients
  - Query status of connections and applications
  - Encode commands & parameters in XML stream

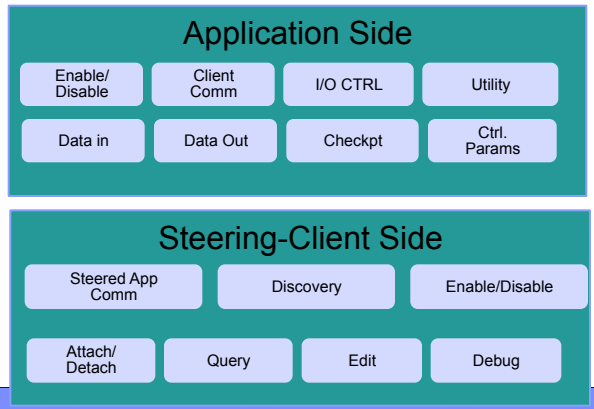
IBM Blue Gene Implementation

- **API implicitly couples IO direction with type of socket connection**
  - i.,e. Output data coincided with a listening socket; Input data coincided with connecting socket
  - Problematic for BlueGene because of limitations of CNK
  - CN (ION) can only establish connecting sockets
- **Solution was to create new API for specifying I/O**
  - Decouple data flow from socket type
- **BlueGene front-end node behind corporate or university firewalls**
  - No direct access to Compute or IO nodes
  - Developed a relay server to allow interaction across firewalls
    - Server also keeps track of steerable applications and the clients that are currently connected
- **Bindings for Python, Perl, and Java**
- **VTK wrappers to abstract the RealityGrid steering interface into a "data reader" class have also been written**
- **Enhanced steering clients: web or native application**

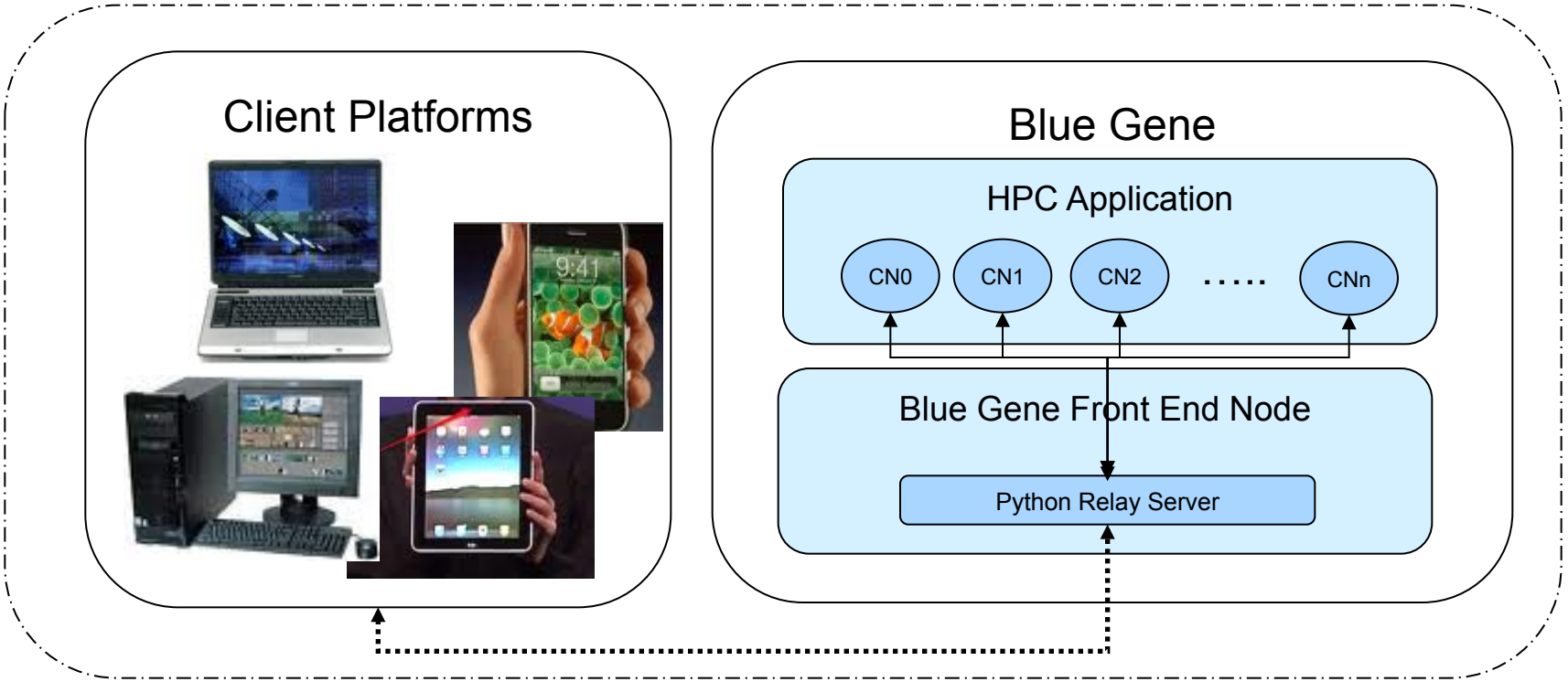
Blue Gene Steering Framework



- **Steering Library Functions** include, but are not limited to:
  - Establish connections between HPC, Steering, Analytics applications running on different platforms across firewalls
  - Register control parameters as read, write, update and transfer values between enabled applications
  - Register data as read, write, update, float, char, integer, etc. and transfer between enabled applications
  - Support multiple connections from different clients
  - Record and transfer checkpoints
  - Query status of connections and applications
  - Encode commands & parameters in XML stream to easily transfer over http connections



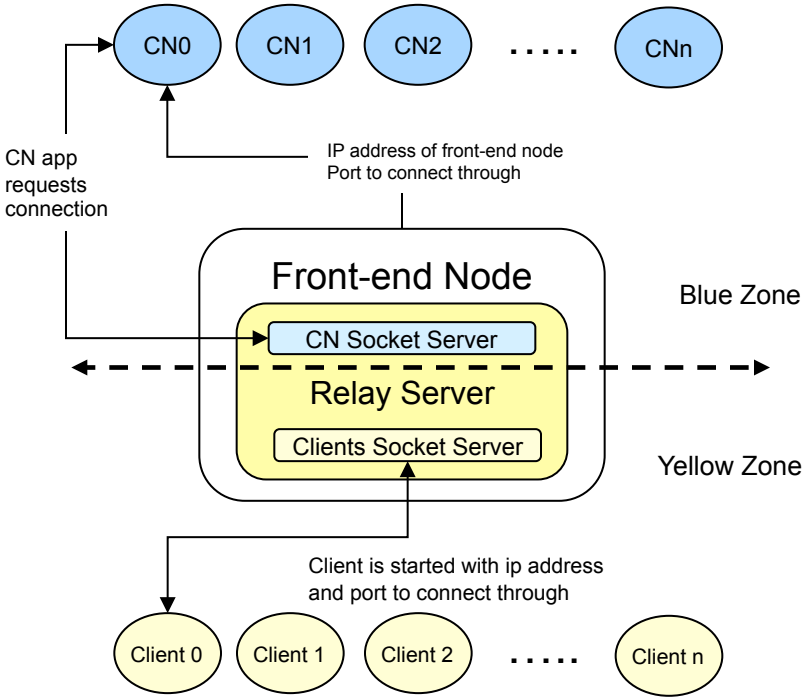
Command and Parameter Flow



- Python Relay Program: Socket Client-Server interface
  - Allows CN to communicate with "outside" world
  - Relay client-side: Establishes in-bound connection with CN socket
  - Relay server-side: Binds connection requests from multiple steering clients
  - Pass thru commands and parameters

Relay Server

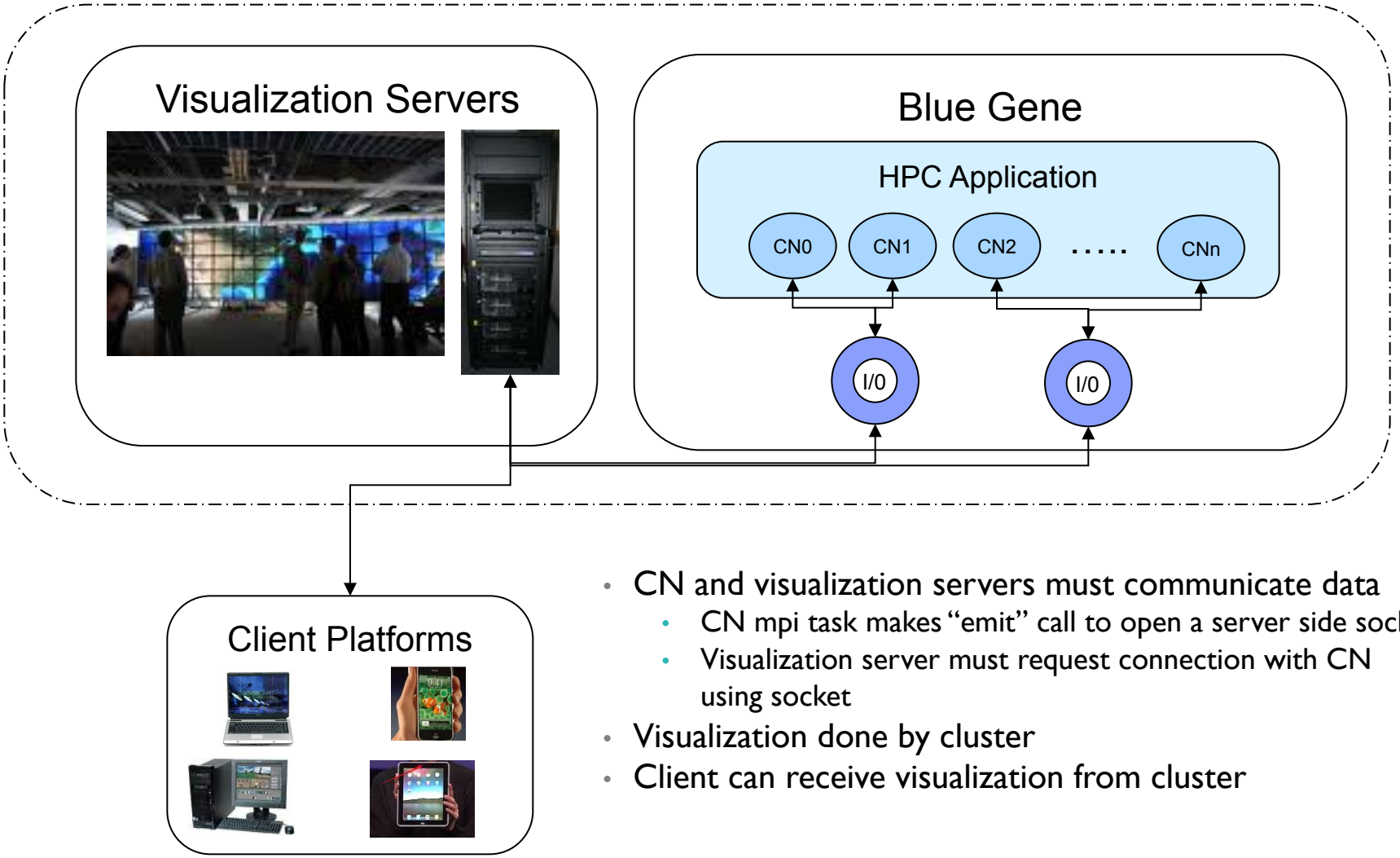
- **Compute Nodes on private network so inbound connections are problematic**
- **One solution is to run relay socket-client on FEN**



Connection Process

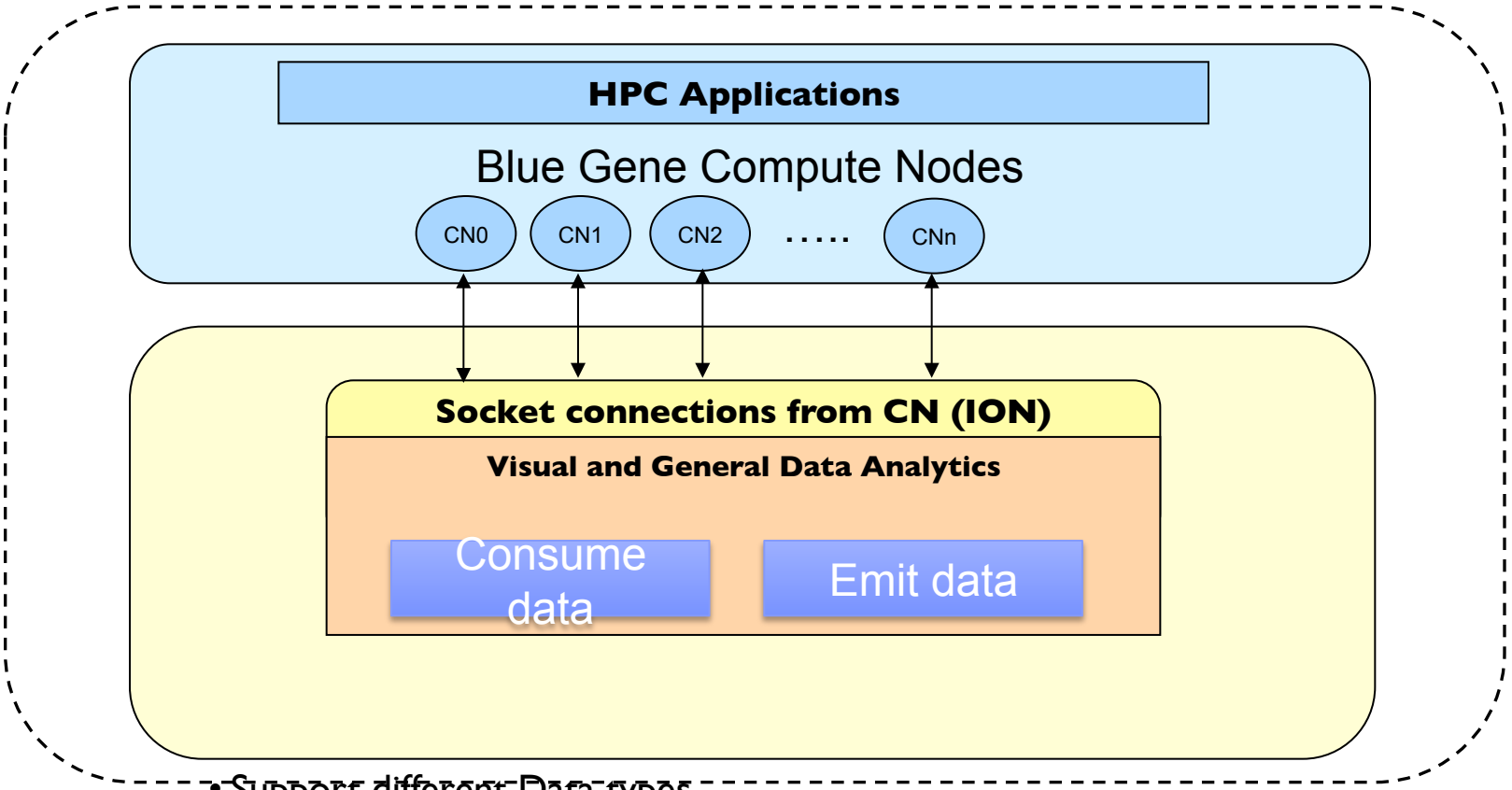
1. CN/ION on private network only accessible by front-end and service nodes
2. Relay server executes on FEN
3. Env variable passed to simulation with ipaddr:port of relay server
4. Simulation connects via socket
5. Client application connects to relay server via socket
6. Connection is established and XML encoded steerable parameters are transmitted from BG to client applications

Command/Parameter and Data Flow



- CN and visualization servers must communicate data
  - CN mpi task makes "emit" call to open a server side socket
  - Visualization server must request connection with CN using socket
- Visualization done by cluster
- Client can receive visualization from cluster

Data Flow: HPC to Visualization Application



- Support different Data types
- Support 2-way flow
- Compute Node to Cluster Node (M:N, where M>=N)
- Visualization platform does NOT have to be co-located with HPC platform,
  - But there must be route from ION to visualization cluster, ie. Same subnet

Basic Application Enablement

Initialization

```
Steering_enable(REG_TRUE);
Steering_initialize(”TEST", numCommands, commands);
```

- **Register the output IO channels**

```
Register_IOType(”TEST DATA",
        REG_IO_OUT,
        REG_IO_CLIENT,
        1,
        &(iotype_handle[0]))
```

- **Register a steerable parameter**

```
status = Register_param("MAX AMPLITUDE", REG_TRUE, (void *)(&amplitude), REG_INT, "1", "15");
 Finalize_registration();
```

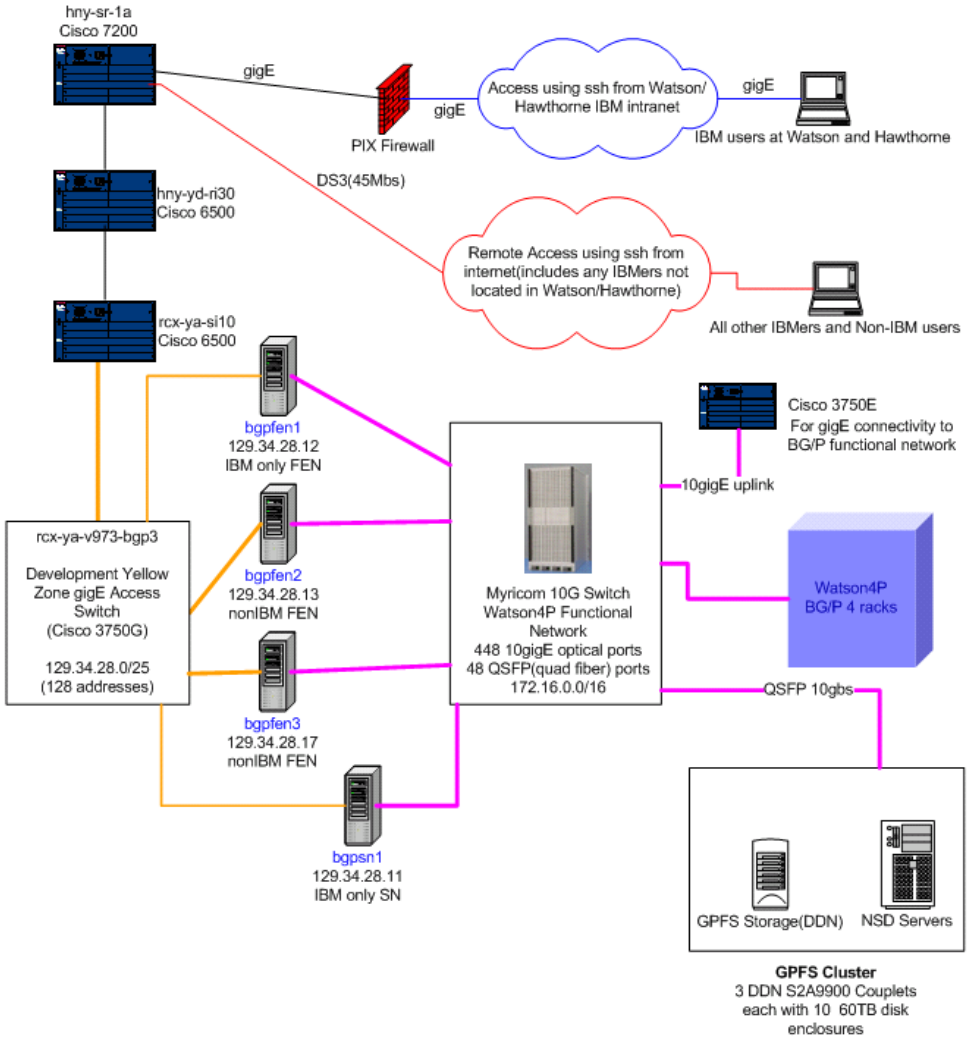
Basic Application Enablement

Main simulation loop

```
While simulating {
    Steering_control(I, &num_params_changed, changed_param_labels,&num_recvd_cmds,
                recvd_cmds, recvd_cmd_params);
    Process recvd_cmds
    for all data to outpu t{
                Emit_start(iotype_handle[j], i, &iohandle)
                Emit_data_slice(iohandle, REG_FLOAT, size,  (void*) data_array);
                Emit_stop();
    }
}
```

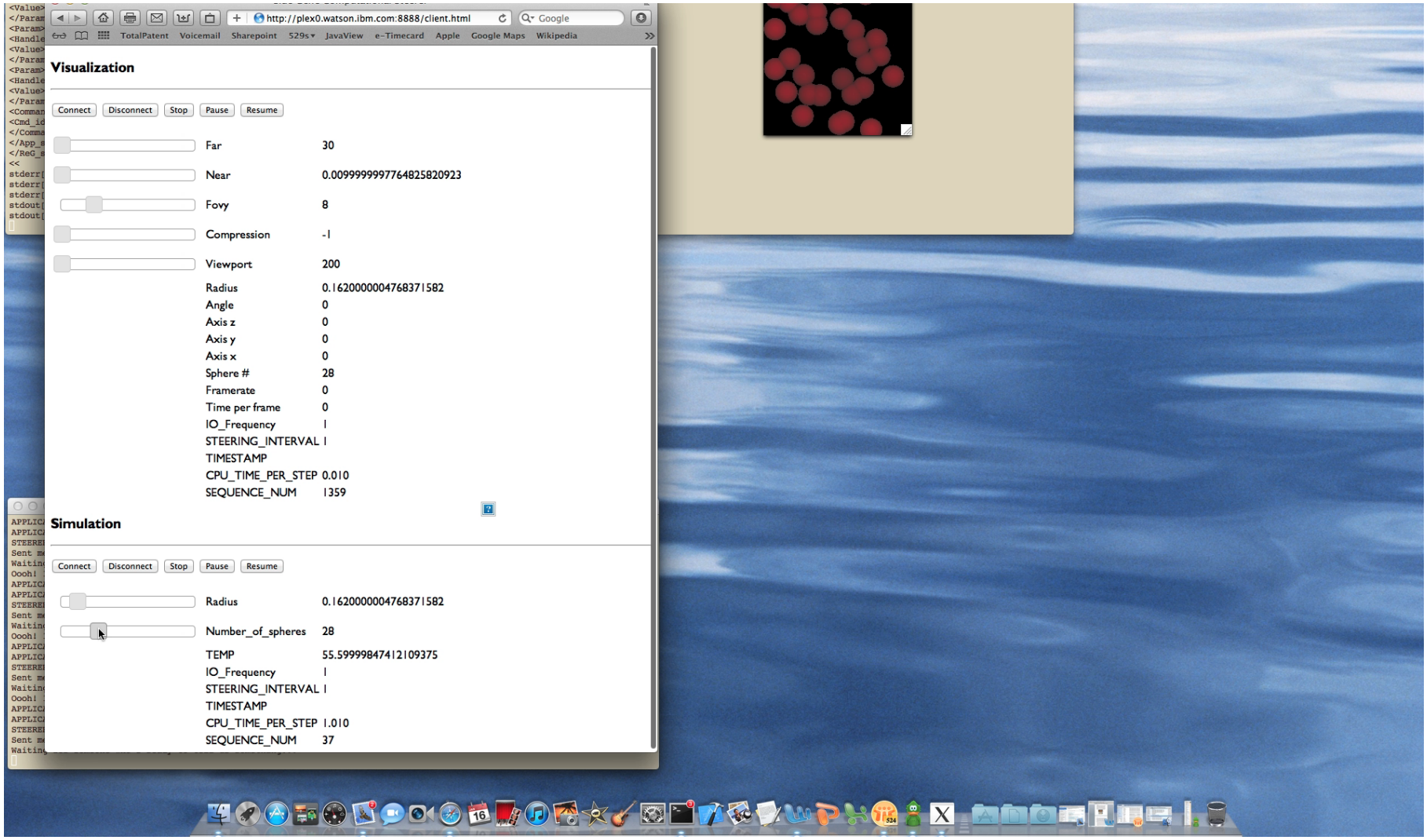
Note: To consume data you replace Emit_start with Consume_start, Emit_data with Consume_datat,etc.

Watson4P Access and Functional Network Overview
Development Yellow Zone Access Design

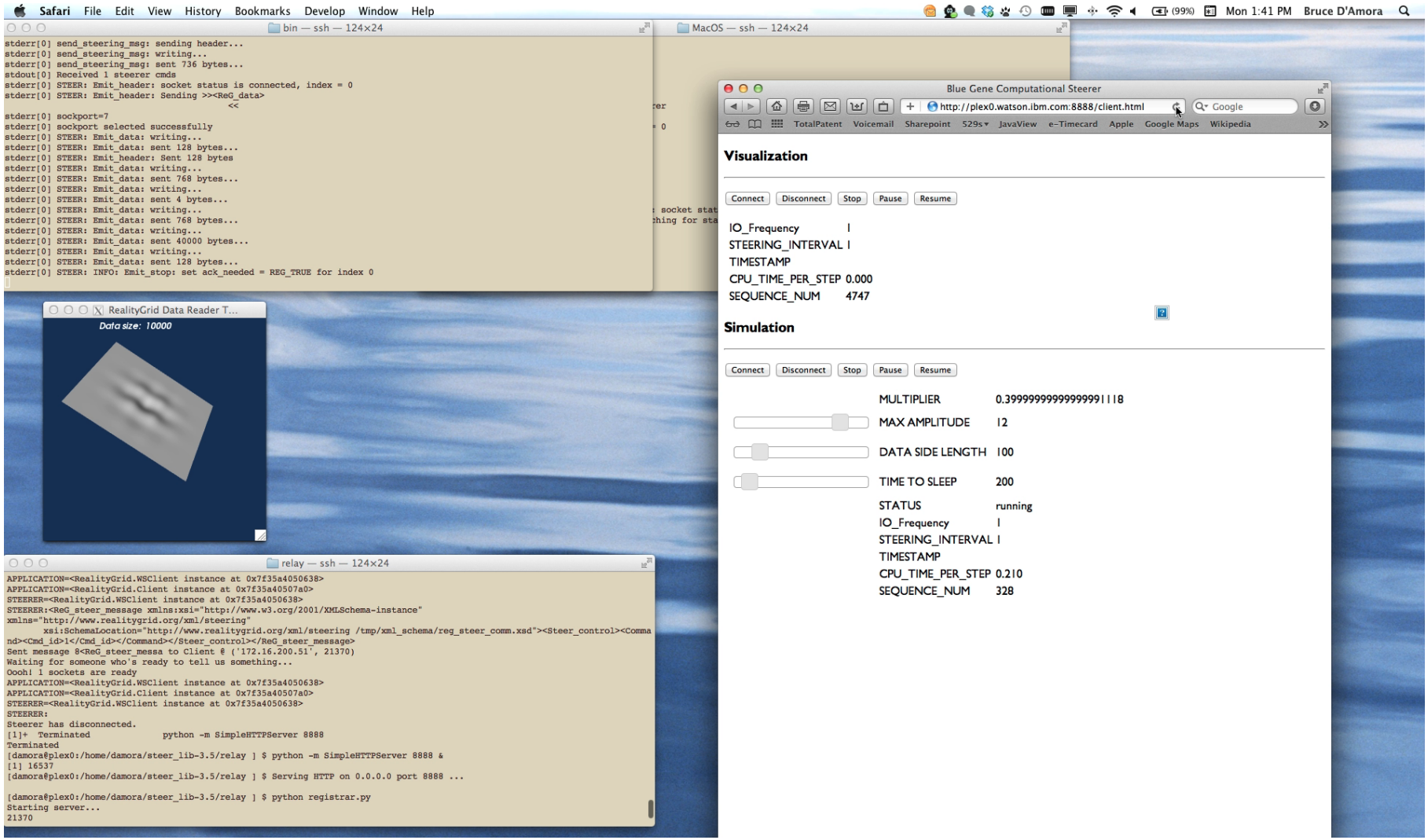


Rev. 3 October 2011

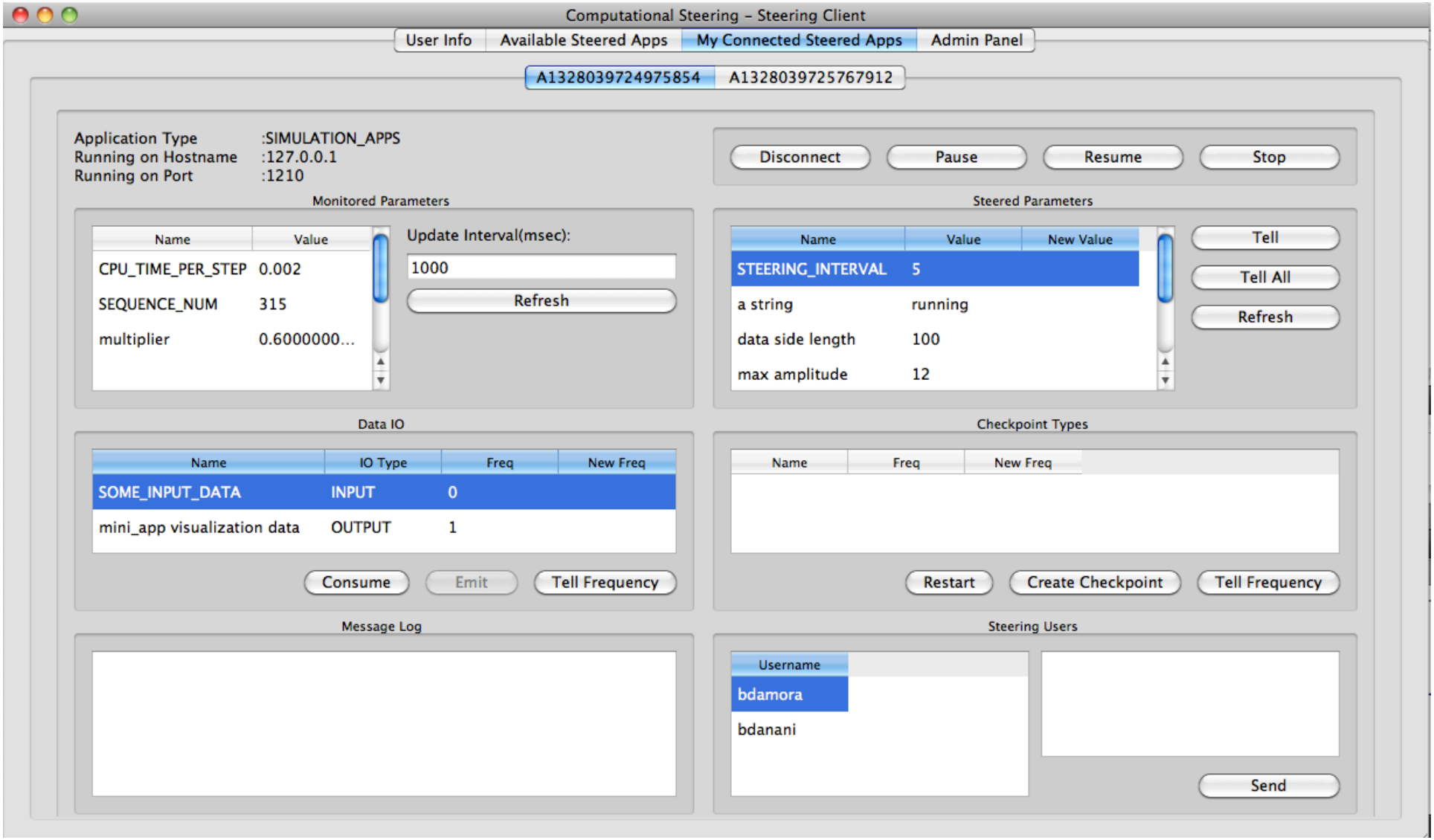
Video



Video



New Steering Client



Next Steps

- **BlueGene/Q enablement**
  - Initial testing, but some bugs remain
- **Fully enable production-level application**
  - OpenFOAM is steerable, but no visualization of output yet
- **Performance characterization and optimization**

Acknowledgements

Thanks for the support of this work go to:

Dinesh Kaushik, David Keyes (KAUST)

Jim Sexton (IBM)