funcXExecutor

Run Parsl Tasks on Globus Compute Endpoints

Name: Ved Kommalapati

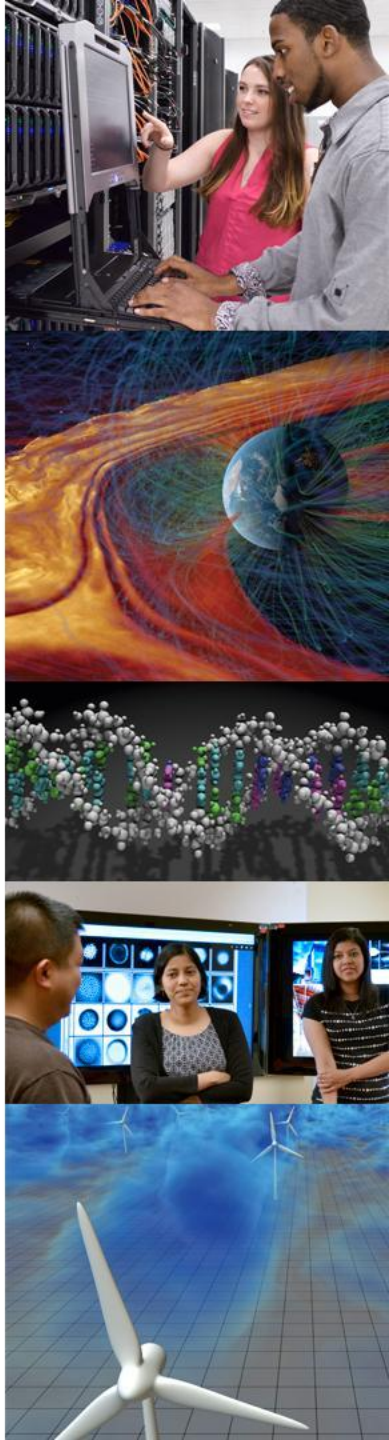
ILLINOIS
NCSA | National Center for Supercomputing Applications

funcXExecutor

- Goal: Create funcXExecutor
  - Allows users to execute Parsl apps on funcX endpoint
  - Provides Infrastructure for Parsl apps on separate endpoints
    - Enables mobile high performance parallel computing
- Methodologies:
  - Development:
    - Currently creating funcX Executor within Parsl
    - Connecting to singular endpoint
    - pytest suite => ensure behavior is consistent
  - Research:
    - Consulted with Ben Clifford

Intended Benefits

- Enhanced Scalability
- Performance
- Fault Tolerance / Load Balancing
- Cost Efficiency
  - Provides Large Compute Resource w/o Single HPC
- Mobile High Performance Parallelized Computing



ILLINOIS NCSA

Current Working Implementation

- Allows users to execute Parsl apps on a singular funcX endpoint
  - Behavior Mostly Consistent w/ Local Thread Execution
  - Observed Unexpected Behavior (TypeError):
    - test_fail.py
      - test_no_deps
    - test_file_apps.py
      - test_files
      - test_increment
- funcX Config File and Executor Implementation
- **Not end goal**
  - Parallelization Amongst Multiple Endpoints / Systems

Current Working Implementation Results

Drawbacks:

- Singular Endpoint Execution
  - Does Not Leverage Multi-endpoint Parallelization
- Limited Parallelization
  - # of Workers Initialized w/ Endpoint

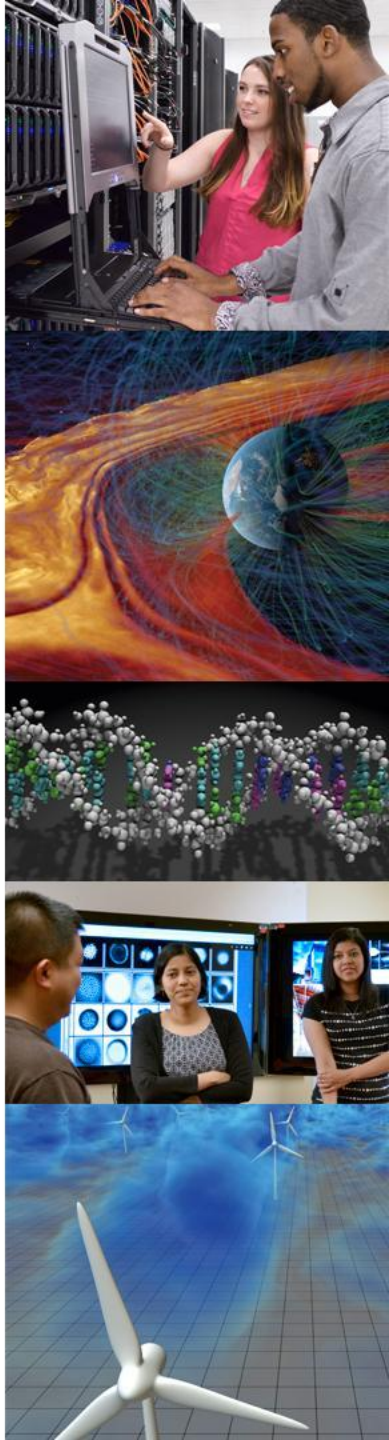
Benefits / Takeaways:

- FuncX Endpoints Can Execute Parsl Apps
- Intersection Between FuncXVisualize and ParslVisualize
  - Research => Understanding Relationships
- Development Experience
  - Exposure to Globus Compute, Parsl, and General Dev. Practices

Multiple Endpoint Implementation

Requirements and Design:

- Pending Tasks (No Endpoint Available)
  - Organized Based on Dependencies (Topological)
  - Data Flow Kernel and Executor Implementations
- Receiving and Interpreting Results
  - Response_Queue, Update Endpoint Pool, Log File
- Endpoint Scheduling System
  - Endpoint Pool
    - Separate Sets of Available and All Endpoints
    - Available => Assigning Tasks
    - All => Endpoint Status Queries

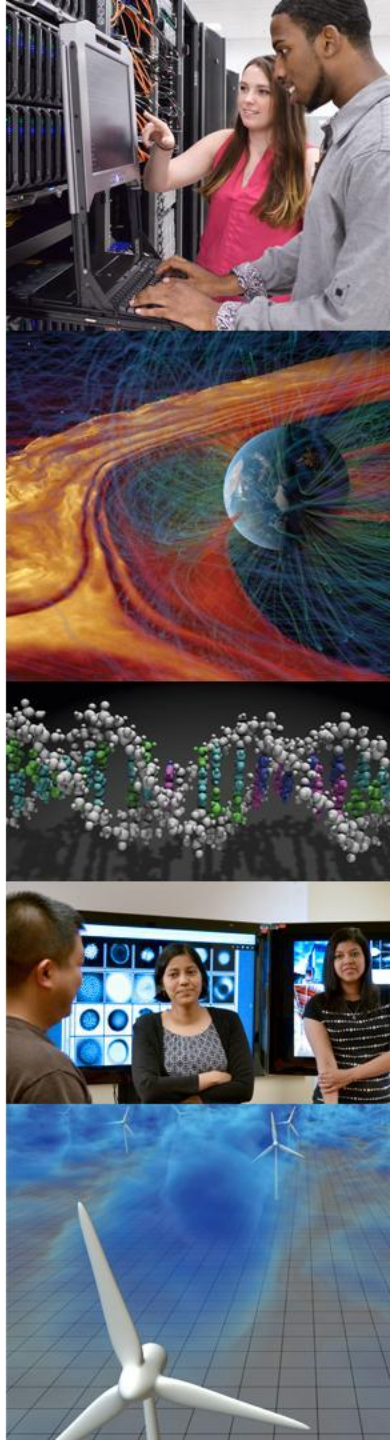


ILLINOIS NCSA

Multiple Endpoint Implementation

Requirements and Design Continued:

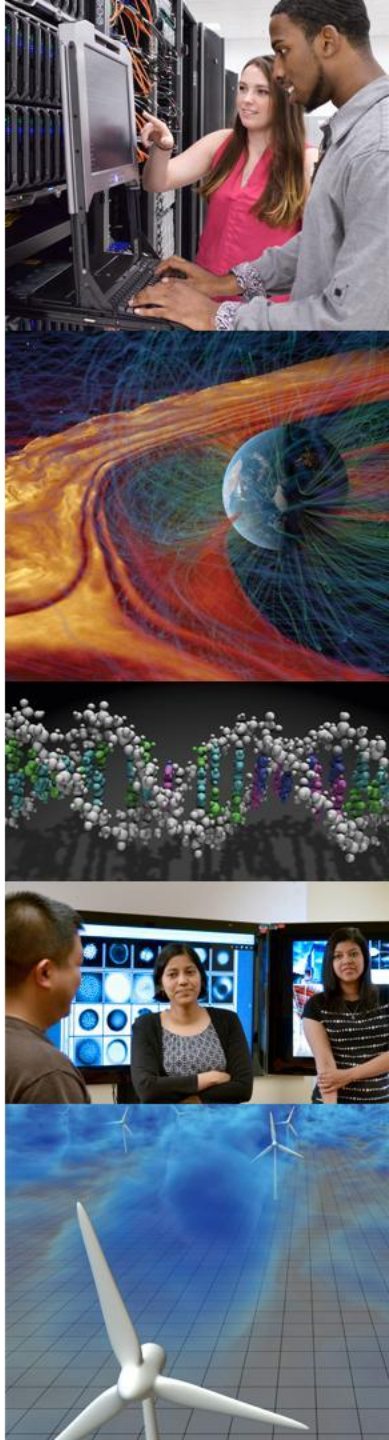
- Modular Parallelization
  - Pre-determined threads_per_endpoint
    - Pre-initialized cores per thread
  - Task Categorization (e.g. Core Requirement)
  - Submit to Thread Closest to Core Requirement
- Race Condition Handling
  - Locks / Mutexes When Updating Lists / Queues
  - Leverage multiprocessing.Queue (Thread Safe)
- **Most Importantly: User Interaction**
  - Params Include User Provided List of Registered Endpoints
  - Discussion Point: Pair Task w/ Core Requirement



ILLINOIS NCSA

Past Roadblocks

- Version Control
  - Uniform Python Version, Conda Environment
- Pytest / Program Execution
  - Importance of Vanilla Terminal
    - VSCode Terminal
      - AMPQS Connection Rejection
      - SSL Certificate Verification Issues
- Leverage Existing Globus Compute Tools
  - Server Self-Diagnostic
  - Monitoring Endpoint Status / Updating Globus
- FuncX Endpoint Information
  - Workers, Cores, Status



ILLINOIS NCSA

Upcoming / Future Objectives

- Flush Out Executor Pool
  - Ensure Optimal Thread Usage Behavior
    - Optional Parameter: Task Core Requirement
- Execution Result Queue / List
  - Context Regarding Execution Status
  - Log File Generation
  - Temporary Return Values => User
    - Associating Return Values w/ Appropriate Futures
    - Optional Parameter: Boolean (Aggregated Value Handling)
    - Temporary Result (Execution Failures)

Ved Kommalapati

- Sophomore at the University of Illinois Urbana-Champaign
- Email:
  - Personal Email: kommalapativ97@gmail.com
  - University Email: vkomm4@illinois.edu
- Slack:
  - Parsl Channel
  - FuncX Channel
- **FuncXVisualize (Prev Project)**
  - Visualizations or FuncX Data Required for Analysis
  - Log File Adjustments
  - Parsl FuncX Intersection

Thank You!

- Dr. Daniel S. Katz
- Mr. Ben Clifford
- Globally Distributed funcX and Parsl Teams
  - University of Illinois
  - University of Chicago
  - Argonne National Laboratory
- NCSA (SPIN)