Streaming Satellite Data to Cloud Workflows for On-Demand Computing of Environmental Data Products

Daniel Zinn[1] Quinn Hart[1] Bertram Ludäscher[1] Yogesh Simmhan[2]

[1]University of California at Davis
[2]University of Southern California

Presented at WORKS 2010

Motivation

- Growing prevalence of sensors, continuously gathering observational data with high spatial and time granularity
- Increased resource needs for storing, managing, analyzing and sharing

→ Investigate Streams as first-class entities in Workflow Systems over heterogeneous resources

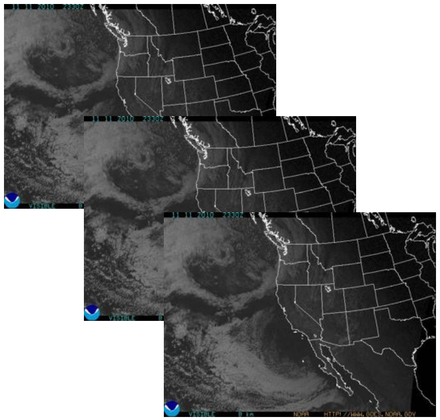
  - What programming abstractions?
  - How to provide these in Cloud/heterogeneous environments?
  - What (new) application areas can benefit from them?

Use-case: Reference Evapotranspiration

- Reference Evapotranspiration (ETo)
  - Planning daily water use (CA farmers, turf managers)
  - Defining water resource policies

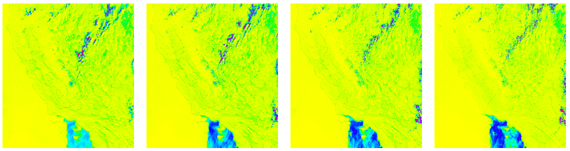
  →improve irrigation scheduling and monitor water stress.
- Current State
  - Single-machine, monolithic 'workflow' orchestrated by make
  - Executed once a day to create ETo maps for CA
- Computed from streaming observational data:
  - Geostationary Operational Environmental Satellite (GOES) (GOES-WEST)
  - California Irrigation Management Information System (CIMIS) stations

Use-case High-level Overview



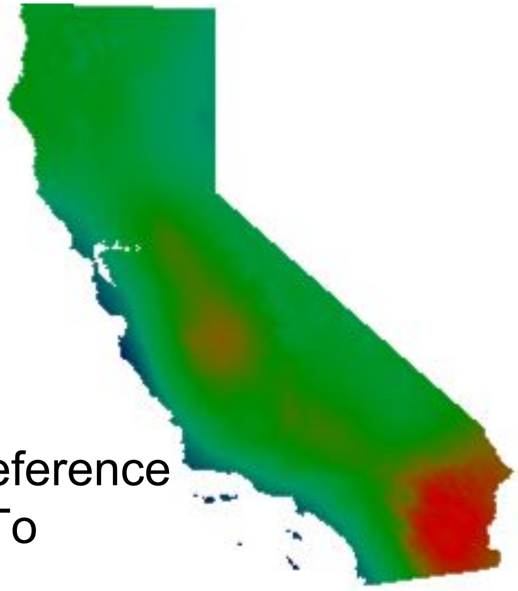
GOES West imagery

CIMIS weather station point data



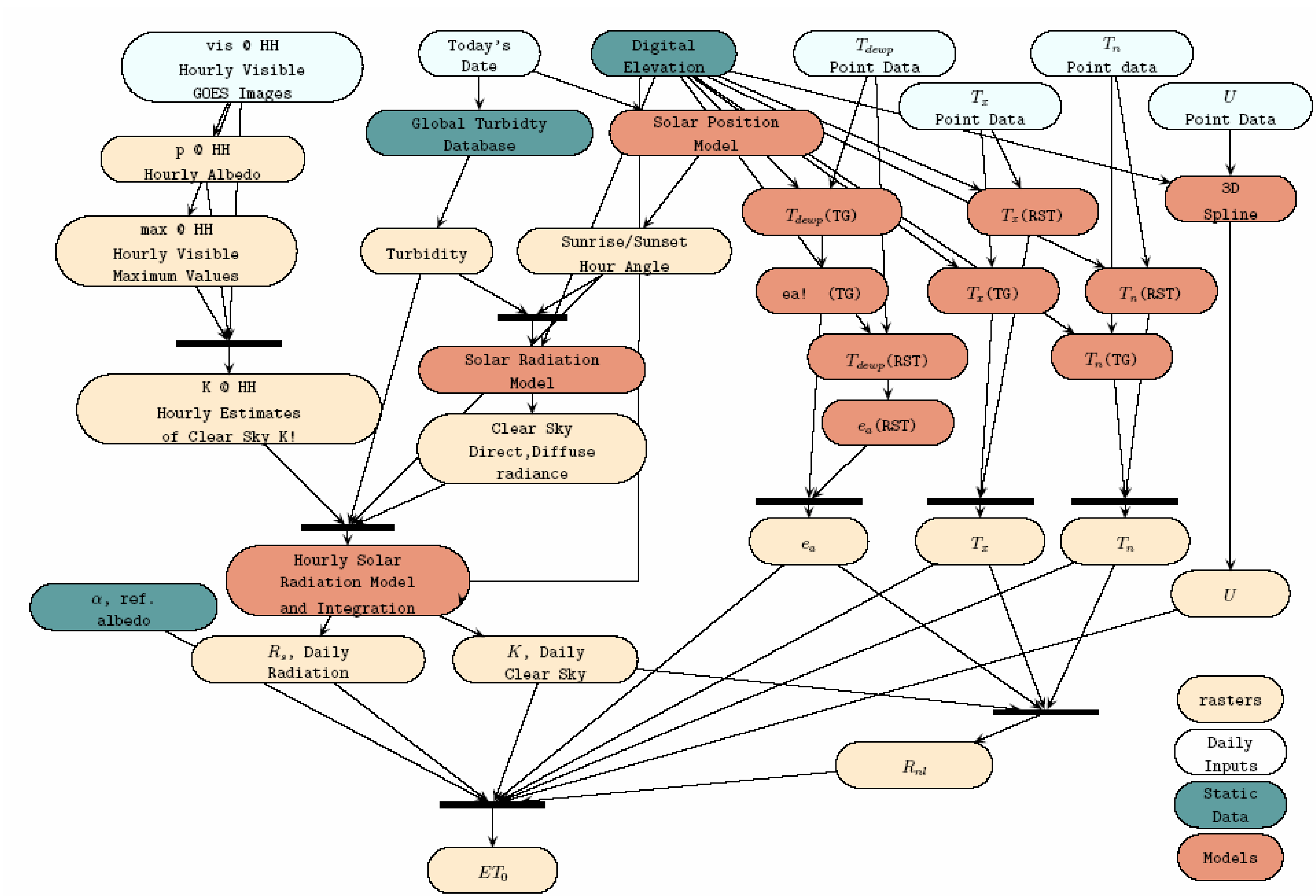
Hourly cloud cover

Complex Makefile Using GRASS GIS



Reference ETo

Use-case Overview



App Domain Requirements

- Need to scale existing applications for
  - Increased data quality [spatial and time]
  - Batch-processing of historic data with novel analysis algorithms
- Need to share computation and data with a large community

- Benefits from Scientific Workflow Technology and Streaming
  - **Streaming** as abstraction maps well to application domain
  - Better management of dataflow pipeline
  - Increased sharing of data and pipeline steps
- Benefits of the Cloud as computational platform
  - Theoretic CPU scale-out limited only by $$$
  - Pay-as you go storage abstractions (BLOB/Tuple store)
  - Simple service abstractions (BLOB, table, worker, …)
  - Relatively easy to access/maintain (vs. GRIDS, vs. owning cluster)

Challenges for Cloud Usage

- Application migration
- Data movement slower / more complex
  - Data-source to cloud
  - Cloud to desktop
  - Desktop to cloud
  - Intra-cloud movement between workers and persistent storage
- Unpredictability
  - Non-homogeneous data transfer rates
  - Non-homogeneous disk access rates
  - Non-homogeneous computation speeds

Contributions

- Migrating legacy script to a cloud-enabled workflow model
- Investigating data movement strategies
  - **BLOB** storage as easy-accessible persistent cloud storage
  - Direct **streaming** from Client to Cloud via TCP socket

  → Streaming faster by a *factor of 5*
- Investigating scale-out behavior
  - Ran 7 workflows in parallel

  → Almost linear speed-up for workflow steps *and* data movement

Workflow Design

- Workflow engine on client, that orchestrates workflow tasks

  Here, we use Restflow workflow system

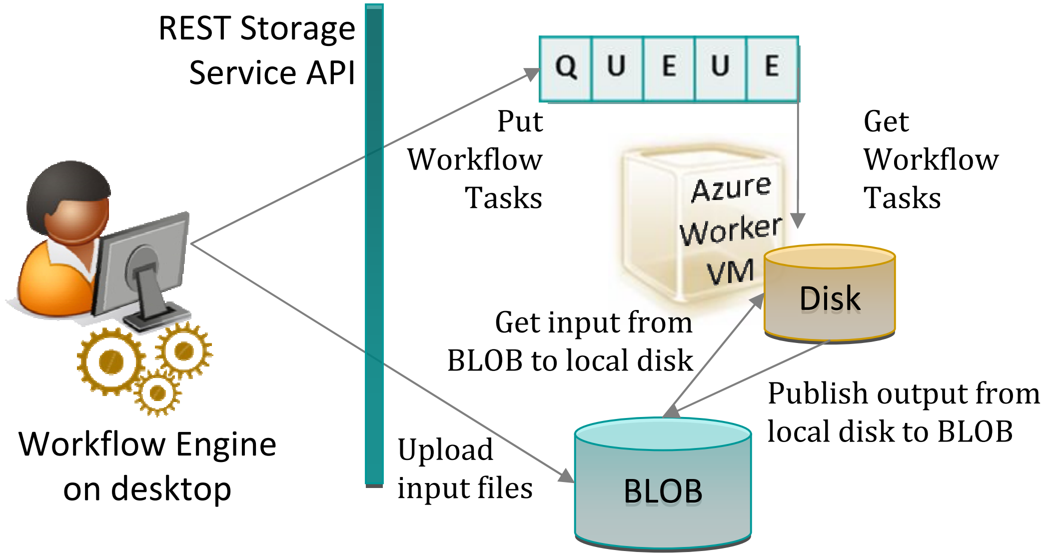
  - Light-weight, Java-based workflow engine with scripting capabilities and automated tracing of data flow and invocation timings

- Workflow tasks run on the Cloud, implement application logics

  Here, we use windows Azure cloud

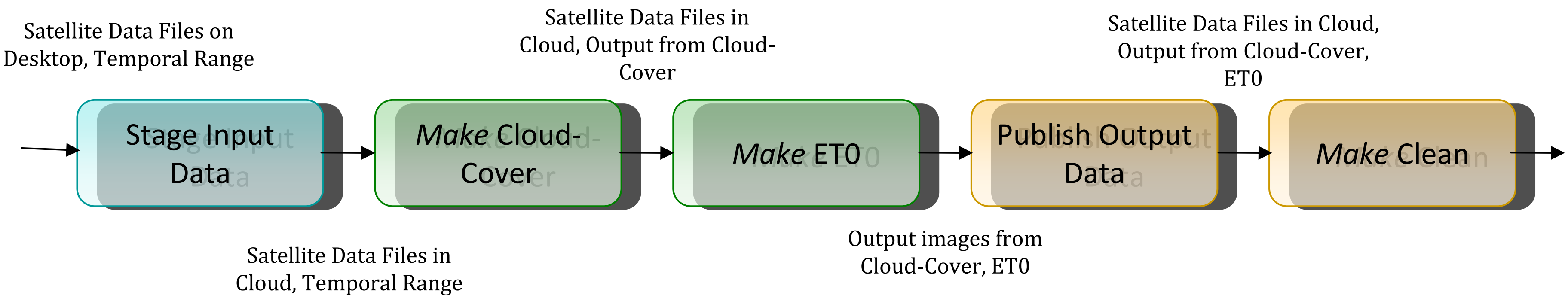
  - We used the PaaS (.Net program) infrastructure
  - Access queues, tables, and BLOB via REST interface

Client – Cloud Communication



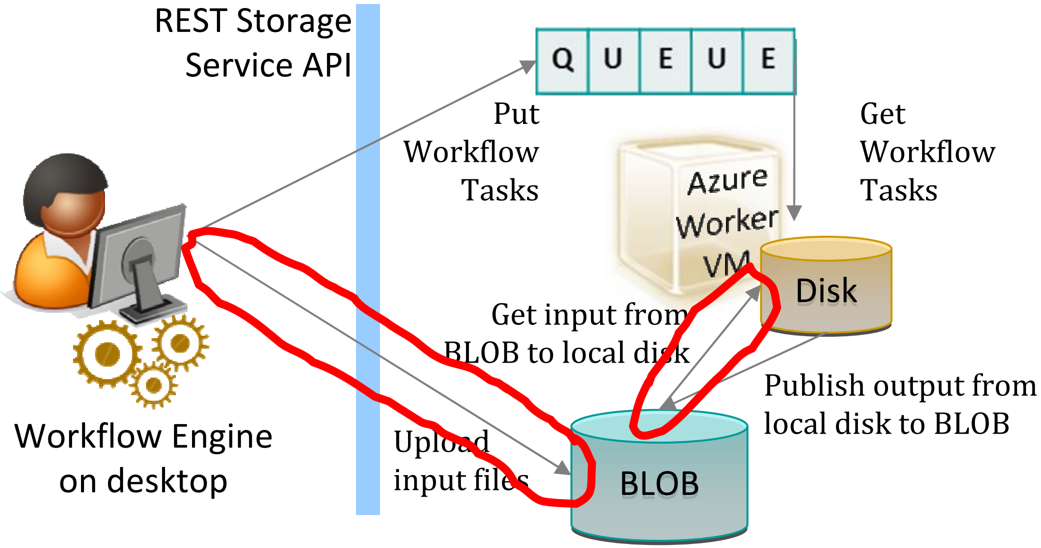
- Instructions via Message queues [Request/Response]
- Input / Output data via BLOB-store
- Workflow tasks are stateful, i.e. locally written files from earlier tasks are re-used by later tasks.
  → all tasks of a single ETo computation are run on the same cloud machine
  → achieved via reserving a cloud machine as first task and switching to private request/response queue

Cloud-Migrating of ETo computation

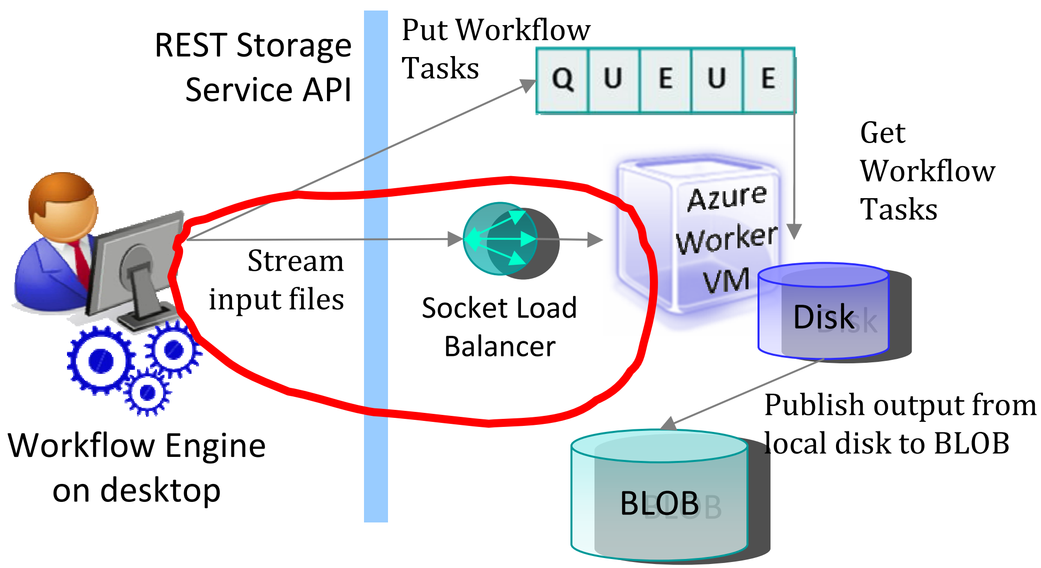


- Stage Input
  - Reserve worker, data up+download via BLOB; 14 days (315MB)
- Make Cloud-cover & Make ETo
  - Re-used legacy Make implementation that calls GRASS, Perl and bash scripts
- Publish Output data
  - Upload output data to BLOB and return URL for user
  - Served by Azure Web-hosting worker
- Make Clean and release worker VM

Improved Data Movement using Streams

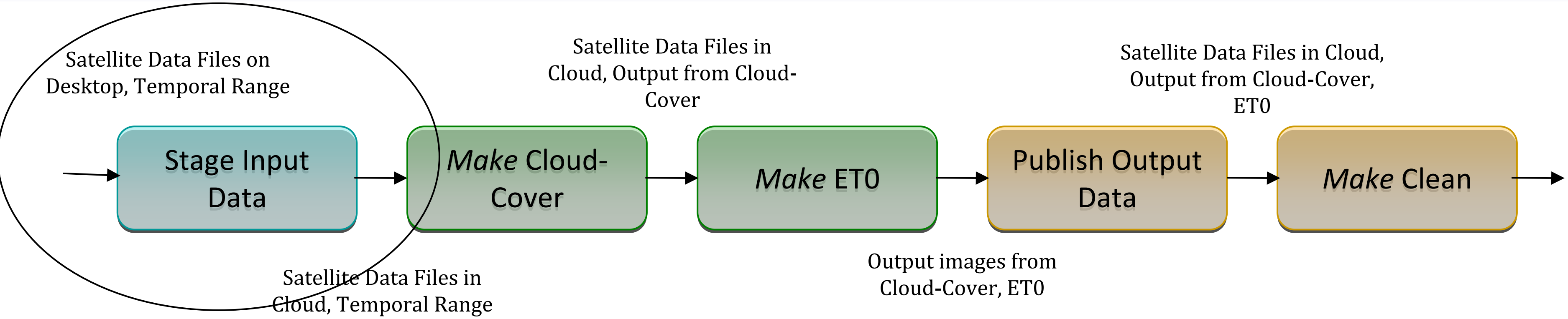


- ☺ Simple API
- ☺ Fault tolerance
- ☹ 2 Data transfers
- ☹ Slower bandwidth



- ☺ Direct transfer
- ☺ Application can work on incoming data incrementally
- ☹ Fault tolerance

Changes for Streaming



- Stage Input

  Reserve worker via "Socket Load Balancer"

  Stream input data to CloudVM directly

- *Rest is unaltered. We still communicate over queues,*

  And store the final output in BLOB for persistency.

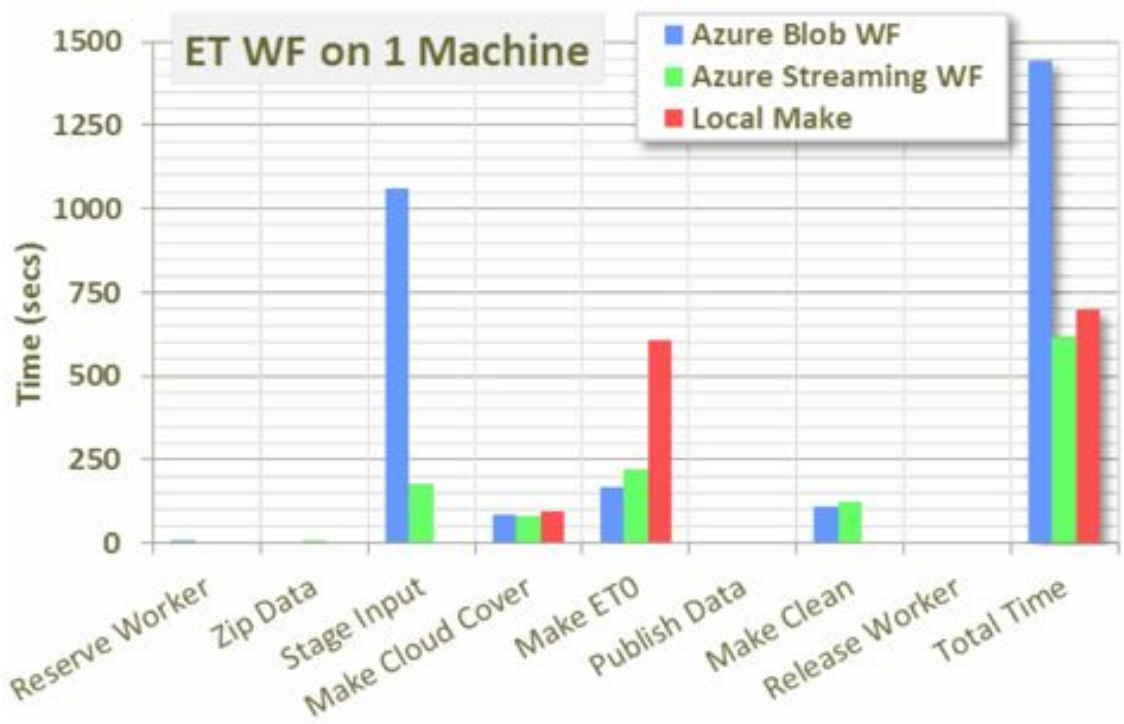
  - Make Cloud-cover & Make Et0
  - Publish Output data
  - Make Clean and release worker VM

Evaluation

- Investigate effectiveness of Cloud Implementation
  - Compare local vs. Cloud execution
  - Investigate Scale-out for 7 concurrent workflows
  - Compare BLOB vs. Streaming strategy
- Experimental Setup
  - Local production machine: 2-core, 2.8GHz, 2GB RAM, NFS-mounted home+data (20MB/s)
  - Client machine: gigabit to the internet
  - Azure worker machine: 1.6GHz, 1.7GB RAM, 250GB local disk space, running 64bit Windows Server 2008. Co-located with data storage account US North Central Data Center
  - Time measured by Restflow actor-invocation tracing capabilities
  - Experiments run 4 times, averages and std-dev reported

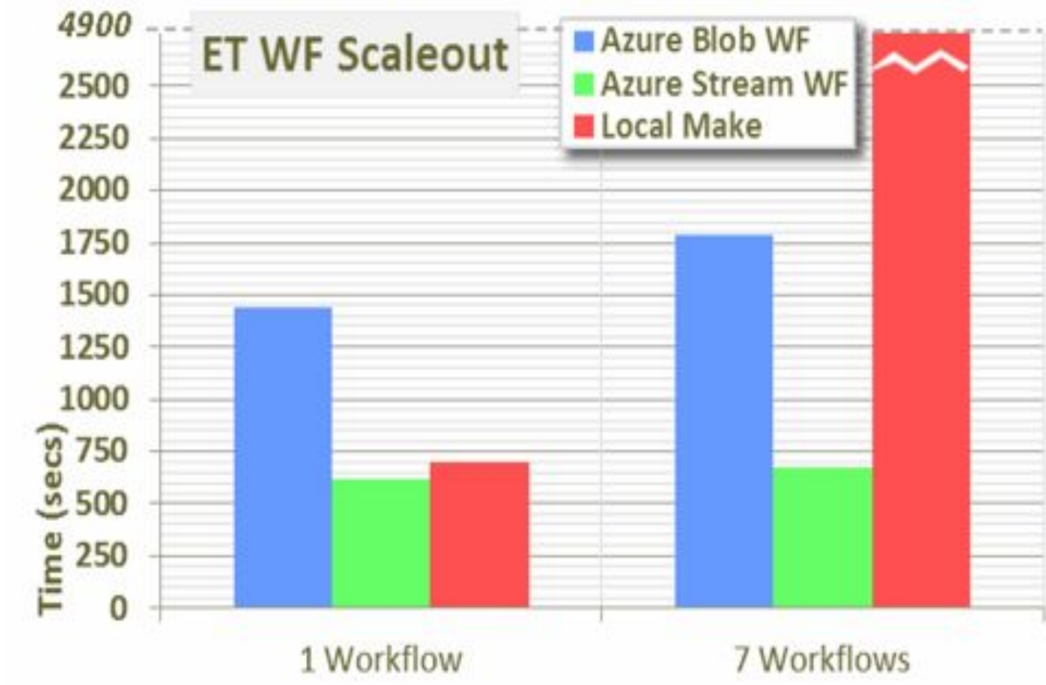
Single Local and Cloud Executions

- Cloud-cover comparable
- ETo performed better on cloud than on local machine (NFS)
- Local outperforms BLOB
- Streaming beats local by 11%



Speedup of Concurrent Cloud Workflows

- Ran 7 workflows in parallel (315MB input data each)
- Linear speedup of computational tasks
- 25% overhead for data movement
  (220s vs. 177s for streaming)
- Overhead for zipping input data on client



Summary:

BLOB: speedup/#proc = 0.80

Stream: speedup/#proc = 0.92

BLOB vs. Streaming Single WF

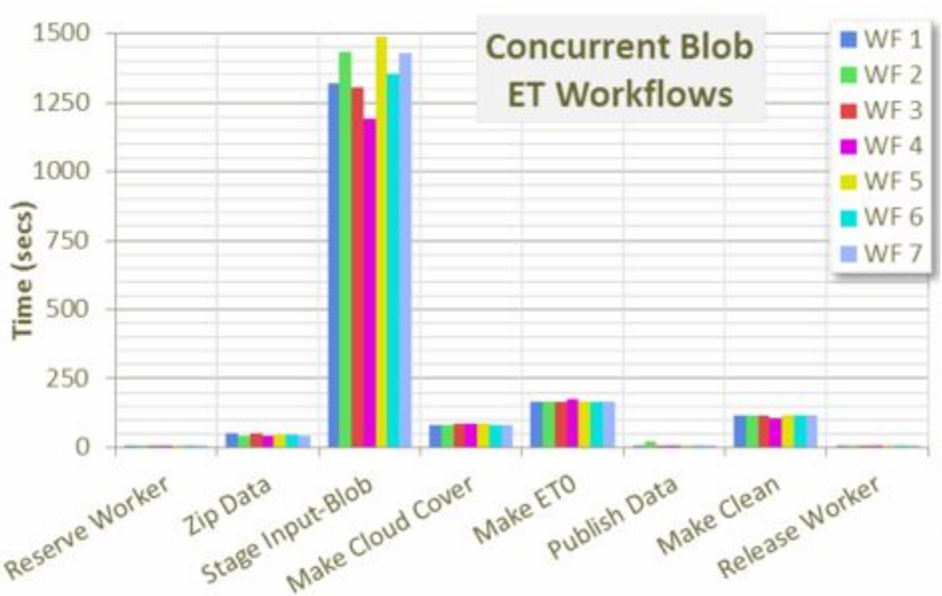
- 315 MB of data in 15 zip files
- BLOB
  - Client->BLOB: 1030s 13% stdev (300KB/s)
  - BLOB->VM: 32s (10MB/s)
- Stream
  - 180s 6%stdev (1.75MB/s)

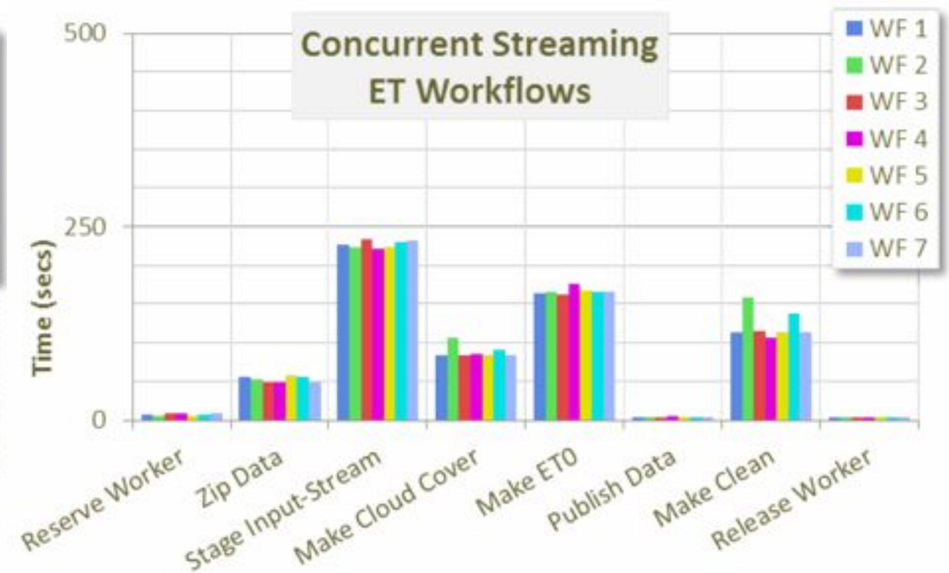
→ Streaming 5x faster, and more stable

BLOB vs. Streaming: Parallel WFs

- BLOB
  - Client->BLOB: 800s—1800s; avg:1172s; (270KB/s); **stdev 44%** *(1.8MB/s total)*
  - BLOB->VM: 32s (10MB/s) **stdev 35%** *(70MB/s total)*
- Stream
  - Avg: 227s 1.35MB/s **stdev 6%** *(9.7MB/s total)*

→ **Streaming faster by >5x and more stable**





→Overall time: streaming 130% / 160% faster (single/parallel)

Findings / Summary

- Migrating ETo to cloud resources feasible from performance-point-of-view
- Streaming paradigm fits application domain well
- Streaming data transport beneficial for performance
  - Much faster than going through BLOB storage
  - More consistent performance
- Streaming on average 5x faster than via BLOB;
  total wall-clock time improvement 130% / 160%

Related Work

C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good.
On the use of cloud computing for scientific workflows.
In *IEEE Fourth International Conference on eScience, 2008. eScience'08*, pages 640–645, 2008.

Gideon Juve and Ewa Deelman.
Scientific workflows and clouds.
Crossroads, 16(3):14–18, 2010.

D. de Oliveira, E. Ogasawara, F. Baião, and M. Mattoso.
SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows.
In *3rd International Conference on Cloud Computing*, pages 378–385. IEEE, 2010.

Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey.
Early Observations on the Performance of Windows Azure.
1st Workshop on Scientific Cloud Computing (ScienceCloud), 2010.

K.R. Jackson, L. Ramakrishnan, K.J. Runge, and R.C. Thomas.
Seeking Supernovae in the Clouds: A Performance Study.
1st Workshop on Scientific Cloud Computing (ScienceCloud), 2010.

Chathura Herath and Beth Plale.
Streamflow-programming model for data streaming in scientific workflows, 2010.
Cluster, Cloud and Grid Computing (CCGrid), 2010.

Ongoing / Future Work

- Compare the impact of streaming on different cloud platforms
- Investigate service abstraction for streams
  - Easy to use such as BLOB, table, queues
  - Built-in fault-tolerance, persistence, and sharing
- Investigating novel streaming apps
  - Energy management (incoming smart-meter streams)
- Collaborative execution of workflows in the Cloud
- Incorporate pay-as-you-go cost model

Thank you.

Questions?