A Fully Automated Fault-tolerant System for Distributed Video Processing and Off-site Replication

George Kola, Tevfik Kosar and Miron Livny
University of Wisconsin-Madison

June 2004

What is the talk about ?

- You heard about streaming videos and switching video quality to mitigate congestion
- This talk is about getting these videos
  - Encoding/Processing videos using commodity clusters/grid resources
- Replicating videos over wide-area network
- Insights into issues in the above process

Motivation

- Education Research, Bio-medical engineering, … have a large amount of videos
- Digital Libraries: Videos need to be processed
  - WCER 'Transana'
- Collaboration => Videos need to be shared
- Conventional approach: Mail tapes
  - Work to load tape into collaborator digital library
  - High turn-around time
- Desire for full electronic solution

Hardware Encoding Issues

- Products do not support tape robot. Users need multiple formats
  - Need a lot of operators!
- Non-availability of hardware miniDV to MPEG-4 encoders
- Lack of flexibility. No video processing support
  - Night video, want to change white balance
- Some hardware encoders are essentially PCs, but cost a lot more !

Our Goals

- Fully electronic solution
  - Shorter turn-around time
- Full automation
  - No need for operators. More reliable
- Flexible software based solution
- Use idle CPUs in commodity clusters/grid resources
  - Cost effective

Issues

- 1 hour DV video is ~13 GB
  - A typical educational research video uses 3 cameras => 39 GB for 1 hour
- Transferring these videos over the network
  - Intermittent network outage => retransfer whole file =>may not complete
- Need for fault-tolerance and failure handling
  - Software/Machine crash
  - downtime for upgrades (we do not control the machines!)
- Problems with existing distributed scheduling systems

Problems with Existing Systems

- Couple data movement and computation. Failure of either results in redo of both
- Do not schedule data movement
  - 100+ nodes, each picking up a different 13GB file
    - Server thrashing
    - Some transfers may never complete
  - 100+ nodes, each writing a 13GB file to server
    - Distributed Denial of Service
    - Server crash

Our Approach

- Decouple data placement and computation
  - =>Fault isolation
- Make data placement full-fledged job
  - Improved failure handling
  - Alternate task failure recovery /Protocol switching
- Schedule data placement
  - Prevents thrashing and crashing due to overload
  - Can optimize schedule using storage server and end host characteristics
  - Can tune TCP buffers at run-time
  - Can optimize for full-system throughput

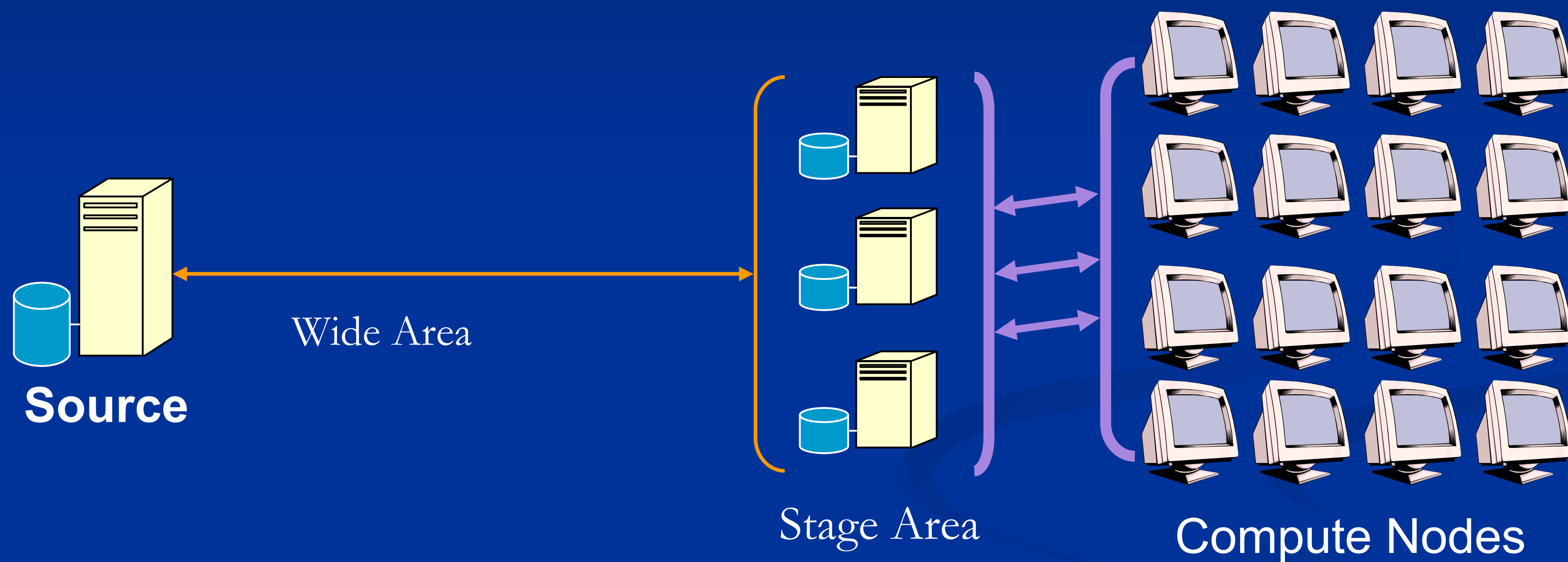
Fault Tolerance

- Small network outages were most common failures
- Data placement scheduler made fault aware and retries to success.
- Can tolerate system upgrade during processing
  - Software had to be upgraded during operation ☺
- Avoiding bad compute nodes
- Persistent logging to resume from whole system crash

Three Designs

- Some clusters have a stage area
  - Design 1. Stage to cluster stage area
- Some clusters have a stage area and allow running computation there
  - Design 2. Run Hierarchical buffer server in stage area
- No cluster stage area
  - Design 3. Direct staging to compute node

Design 1 & 2: Using Stage Area



Design 1 versus Design 2

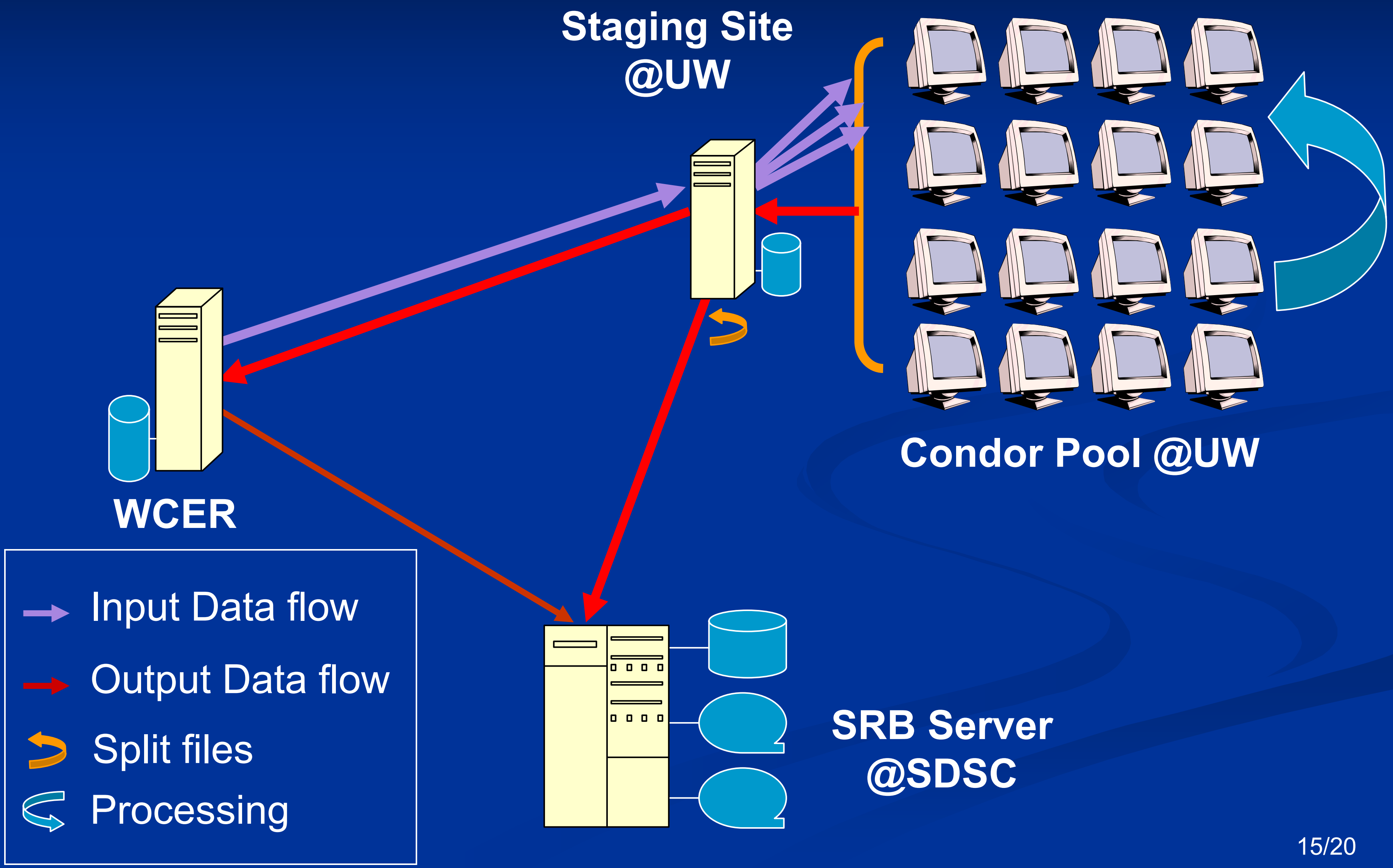
- Design 1
  - uses the default transfer protocol to transfer data from stage area to compute node
  - Not scheduled
  - Problems when number of concurrent transfers increases
- Design 2
  - uses a hierarchical buffer server at the stage node. Client runs at the compute node to pick up the data
  - Scheduled
  - Hierarchical buffer server crashes need to be handled
  - 25%-30% performance improvement in our current setting

Design 3: Direct Staging

Design 3

- Applicable when there is no stage area
  - Most flexible
- CPU wasted during data transfer/Need additional features
- Optimization possible if transfer/compute times can be estimated

WCER Video Pipeline



WCER Video Pipeline

- Data transfer protocols had 2 GB file size limit
  - Split files and rejoin them
- File size limits with Linux video decoders
  - Picked up new decoder from CVS
- File system performance issues
- Flaky network connectivity
  - Got network administrators to fix it

WCER Video Pipeline

- Started processing in Jan 2004
- DV video encoded to MPEG-1, MPEG-2 and MPEG-4
- Has been a good test for data intensive distributed computing
- Fault tolerance issues were the most important
- In a real system, downtime for software upgrades should be taken into account

Encoding	Resolution	File Size	Average Time
MPEG-1	Half (320 x 240)	600 MB	2 hours
MPEG-2	Full (720x480)	2 GB	8 hours
MPEG-4	Half (320 x 240)	250 MB	4 hours

How can I use this ?

- Stork data placement scheduler
  - http://www.cs.wisc.edu/stork
- Dependency manager (DAGMan) enhanced with DaP support)
- Condor/Condor-G distributed scheduler
  - http://www.cs.wisc.edu/condor
- Flexible DAG generator
- Pick our tools and you can perform data intensive computing on commodity cluster/grid resources

Conclusion & Future Work

- Successfully processed and replicated several terabytes of video
- Working on extending design 3
- Building a client-centric data-aware distributed scheduler
- Deployment of the new scheduler inside existing schedulers
  - Idea from virtual machines

Questions

- Contact
  - George Kola kola@cs.wisc.edu
  - http://www.cs.wisc.edu/condor/didc