NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

BERKELEY LAB

MDC2- testing
Multiple Event-Components

http://www-rnc.lbl.gov/GC

http://gizmo.lbl.gov/sm

LBNL/ANL/BNL

GC HENP Collaboration

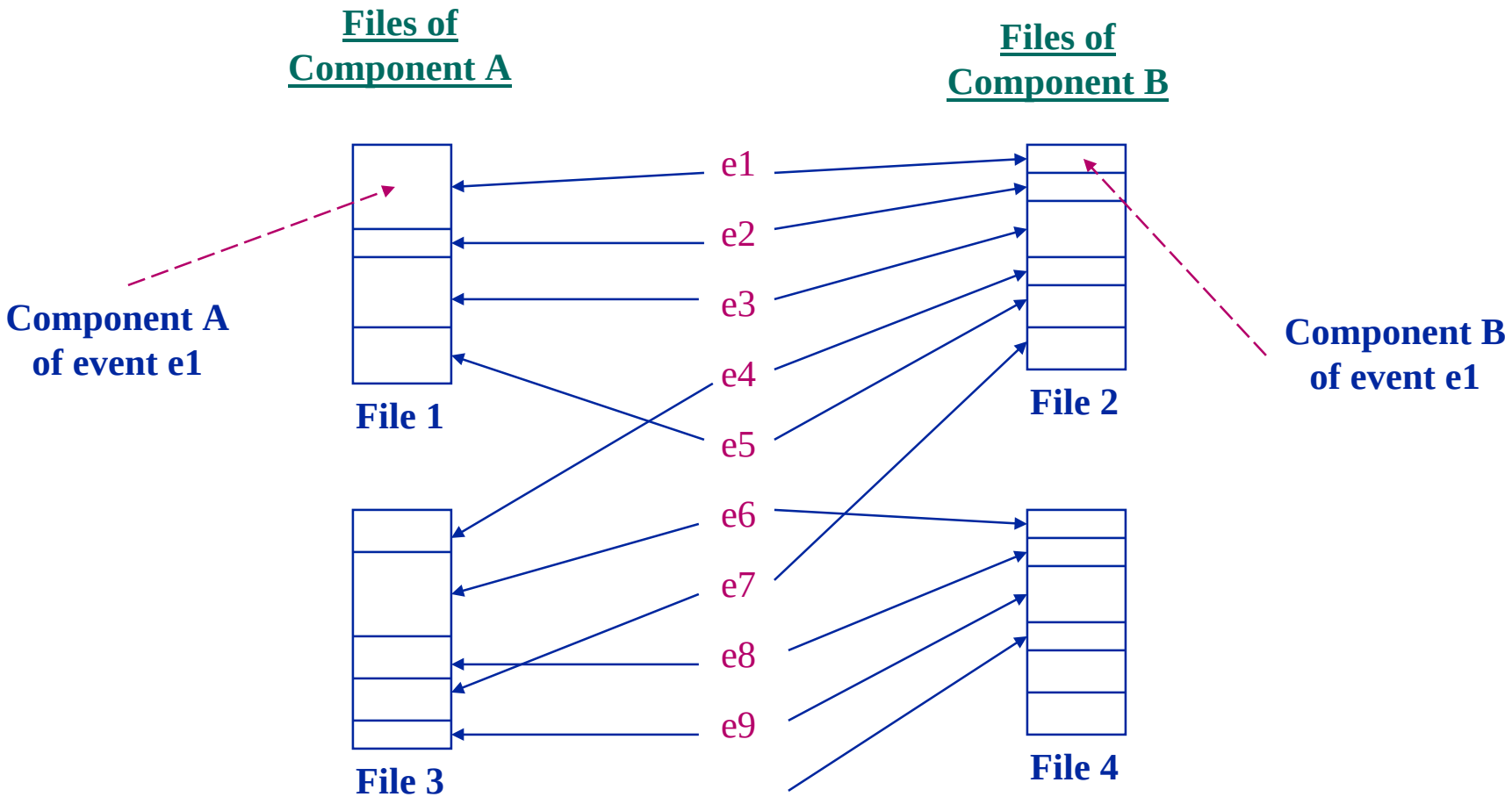
April, 1999

Multiple Event-Components

- **Event Components**
  - **partition each event into 5-10 pieces**
  - **tracks, hits, vertices …**
- **Queries can request one or more components**
  - **all components of an event must be in disk cache at the same time**
- **Problem: how to manage multiple component files to minimize re-caching of files**
- **Pseudo query language**

```
SELECT tracks, hits
FROM Run17
WHERE glb_trk_tot>0 & glb_trk_tot<10 &
n_vert_total<3
```

Example of multiple components



File Bundles: (F1,F2: e1,e2,e3,e5), (F3,F2: e4,e7), (F3,F4: e6,e8,e9)

File Weight Policy for Managing Multi-Component-Events

- File weight (bundle) = 1 if it appears in a bundle,
  = 0 otherwise
- Initial file weight = SUM (all bundles for each query) over all queries

$$IFW = \sum_{queries} \sum_{\substack{all \\ bundles}} file_weight_per_bundle$$

- Dynamic file weight: the file weight for a file in a bundle that was processed is decremented by 1

$$DFW = \sum_{queries} \sum_{\substack{all \\ bundles}} file_weight_per_bundle$$

$$- \sum_{queries} \sum_{\substack{processed \\ bundles}} file_weight_per_bundle$$

Caching Policy (1)

- **Query service policy**
  - round robin
  - query is skipped if no bundle fits available cache
  - when query is skipped, a skip_service counter is incremented.
  - if counter is above preset limit, service to other queries stops till this query is serviced

Caching Policy (2)

- **Bundle caching policy**

$$Bundle_weight = \sum_{\substack{all\ files \\ in\ the\ bundle}} dynamic_file_weight$$

  - select bundle with most files in cache
  - if a tie, select bundle with highest weight
  - if not enough space in cache, select next bundle that fits in cache
  - if none fit in cache, select next bundle with one less file in cache, etc.
  - if no bundles found, skip query, and increase skip_service counter

Caching Policy (3)

- **File purging policy**
  - files are in 2 categories:
    - file currently in use
    - file not currently in use
  - purge file not_in_use with lowest dynamic_file_weight
  - if a tie, purge largest file

- **Pre-fetching policy**
  - Initially: unlimited
  - this parameter can be assigned dynamically

Experiment Setup

- Hijing simulation
- 930 files, 24,000 Events
- 8 Components, NO multi-bundle overlap
  - dst, EbE, HIT, RAW, RAW1, RAW2, Str, TRK
- file size varies:
  - 110 MB, 6 MB, 15 MB, 1.17GB, 6 MB, 22 MB
- properties: tagrand, fid.dst, …
- query examples:
  - "20000<=fid.dst<=20050 & tagrand>1000.0" : "dst;RAW1;RAW2;" : .1 : 5
  - "20010<=fid.dst<=20108 & tagrand>1000.0" : "dst;RAW1;RAW2;" : .1 : 5
  - "20090<=fid.dst<=20150 & tagrand>1000.0" : "dst;RAW1;RAW2;" : .1 : 5

Test Goals

- Robustness - check if crashes
- Handle Interruptions - by other systems (HPSS, Objectivity)
- Correctness - does what's expected
- Work with UC
- Work with UC on Linux
- Multi-drive efficiency
- Efficiency measurements
  - file overlap by events
  - file overlap by components
  - slow down of small query by large "background" queries

Tracking Files and System Performance

Conclusions / Observations

- **Robustness**
  - **No crashes, except when:**
    - **tried to fix code dynamically on QM**
    - **ran 39 parallel queries: worked with QM at LBNL (run 25)**
    - **objectivity lock server was down**
  - **Actions:**
    - **need to coordinate patch level between LBNL & BNL**
    - **in future will avoid depending on Objectivity for getting file names. Will use file name index.**

Conclusions / Observations

- **Handle interruptions**
  - **worked OK when HPSS was down (run 2)**
  - **worked incorrectly when went over HPSS file limit (run 30)**

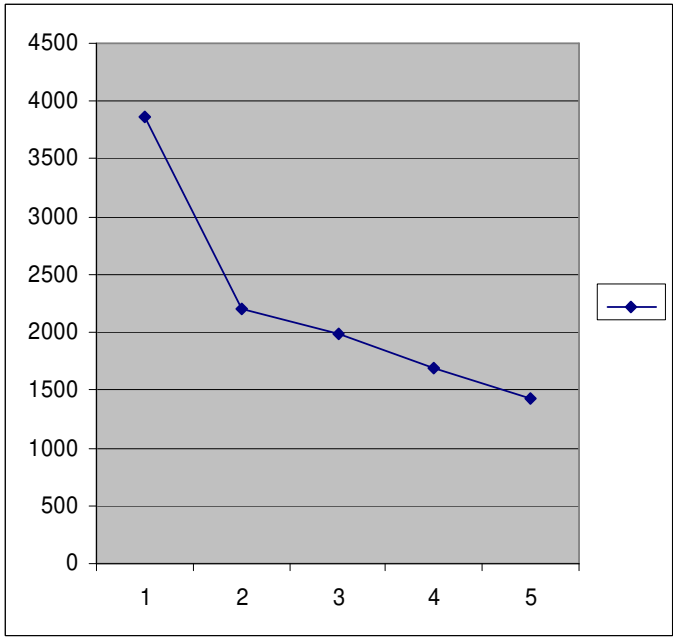
  – **Actions:**
  - **CM will keep a queue of PFTP requests if HPSS refuses**
  - **CM will schedule another PFTP after one completes**
  - **CM will impose limit on PFTPs issued based on config file parameter (to be discussed)**
  - **In case of HPSS "no response", CM will retry periodically till it comes back up**
  - **QM is oblivious to all that.**
  - **QM will send "abort file PFTPs" request if needed.**
  - **QM will handle "file cached" even when query aborted.**

Conclusions / Observations

- **Correctness**
  - Several unexpected behavior in QM (runs 2,3,5)
  - Most problems traced to race conditions
    - need to lock data structures - fixed
  - Problem also traced to skipping queries bundle requested too early (run 19)
    - fixed
- **Work with UC**
  - fails initially (wrong setup), but then OK (run 21)
- **Work with UC on Linux**
  - could not perform, because Objectivity lock server down

Multi-drive efficiency

- **Multi-drive efficiency (runs 8-13)**
  - ran 7 components in query in order to have files in 5 tapes (was hard to find)
  - results:
    - 1 drive - 3865
    - 2 drives - 2200
    - 3 drives - 1993
    - 4 drives - 1669
    - 5 drives - 1432



Observations:

- 1 drive: transfer rate intermittent
- 2 drives: transfer rate sustained 5-6 MBs
- 3-5 drives: drives idle a lot

Conclusion: could use more pre-fetching

Efficiency Measurements

- file overlap by events (runs 3-4, 19-20)
  - same query run twice 15 min delay
  - about 40% improvement
- file overlap by components (runs 27-28)
  - 3 queries ran with no delay (best for no policy)
  - no policy ran slightly better
  - cache was small, pre-fetching “hogged cache”
  - put in “smart pre-fetching” - no pre-fetching if cache small
  - new results: policy 10% better (still no delay between queries)
  - **Conclusion**: need to consider “pre-fetching with preemption”

Efficiency Measurements

- **slow down of small query by large "background" queries (runs 29-30)**
  - **background: 36 bundles x 7 components = 252 files**
  - **short query: 4 x 3 = 12 files**
  - **short query standalone: 375 sec**
  - **short query "injected": 530, 643, 548 sec**
  - **short query "injected" quickly: 41 sec**