SAM for Intensity Frontier Experiments

Robert Illingworth
Fermilab/CD

11 November 2009

SAM Data Handling System

- Data handling system originally developed for D0 Run II
  - Later adopted by CDF and Minos
- Centralized database containing file metadata and location catalogue
- File movement and process tracking handled by a component called a “station”
- SAM is supported by the REX department

SAM metadata

- Metadata includes things like file name, size, run number, run type, first and last event numbers, file type, application and version used to create it, etc
- Derived files have a list of parent files used to create the file
- Also “parameters”: arbitrary name, value pairs

- Metadata is queried by “dimensions”: simple query language used to select files by their metadata values

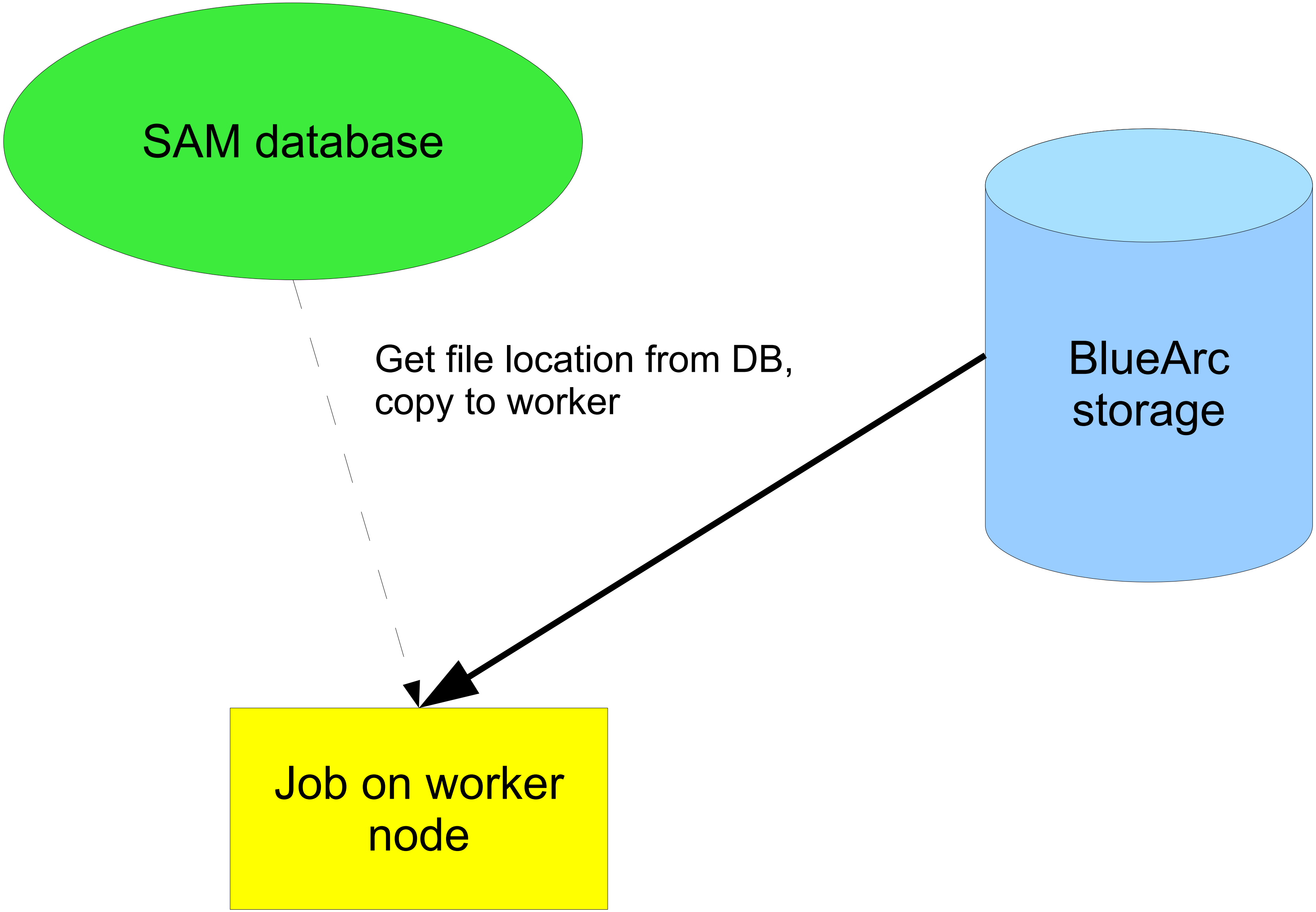
Usage scenarios

- Will assume that entire experiment dataset is available from centrally mounted, shared, storage area with POSIX file access (ie BlueArc)
  - tape used only for archive/backup
- Also assume that access to the data is available via gridftp servers or similar where the centrally mounted area is not available

Usage scenarios - local

- Simplest
  - Use metadata and location catalogue
  - Experiment framework (or pre-execution script) responsible for creating a list of file names and obtaining their physical locations
  - If the load on the central storage needs to be limited, then need some external method to do it
  - Easy for everybody to understand

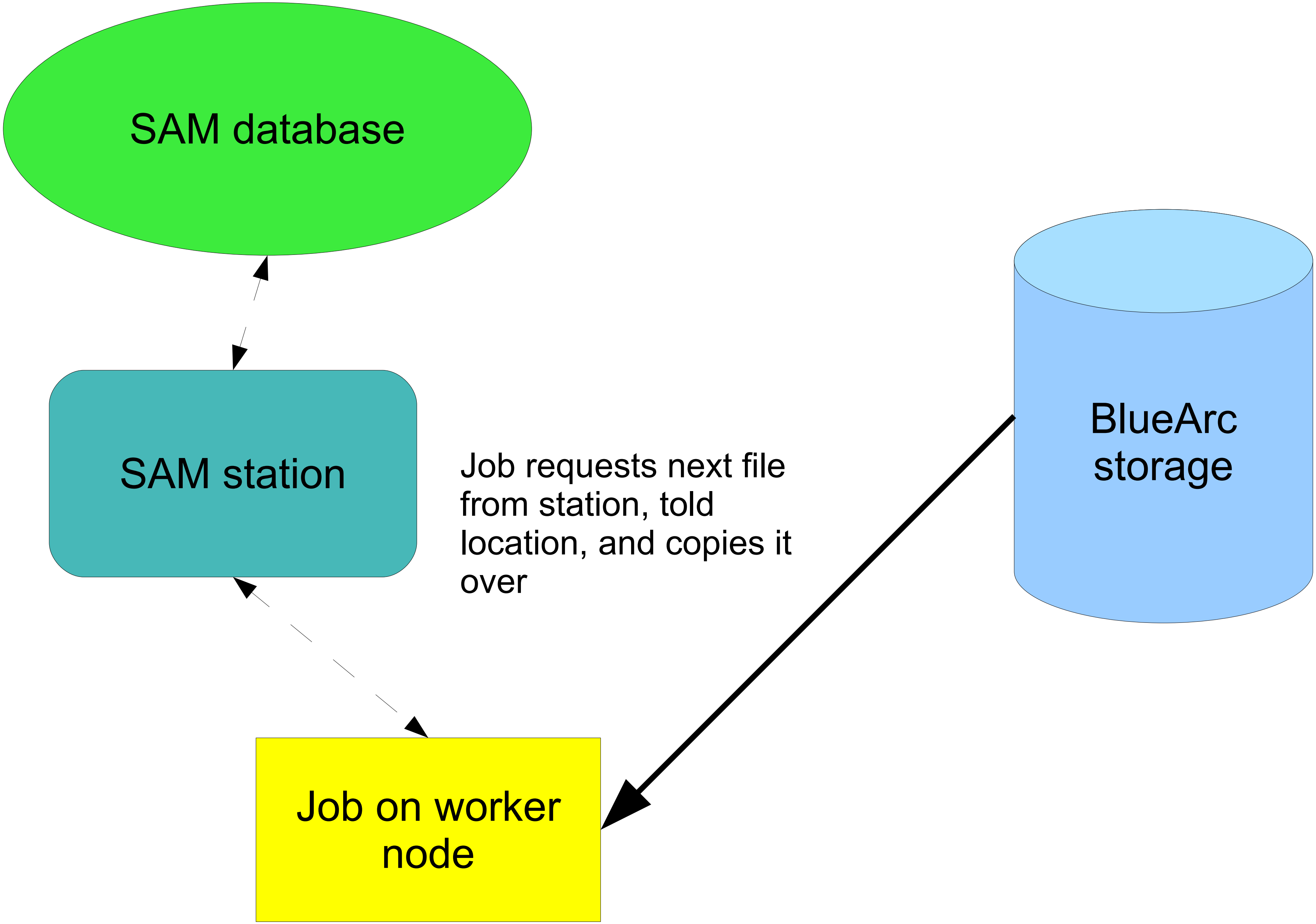
Usage scenarios - local



Usage scenarios - local

- Add a SAM station
  - No cache – direct access from BlueArc
  - Start a SAM project by giving it a dataset definition
  - Framework asks project for files – project returns name and location of next file (in arbitrary order)
  - Framework is responsible for copying file
    - SAM station could provide load control, but not sufficient on its own
  - More complicated
  - Use SAM database to track which files processed
  - Can run multiple jobs per project; divide the load between them automatically

Usage scenarios - local

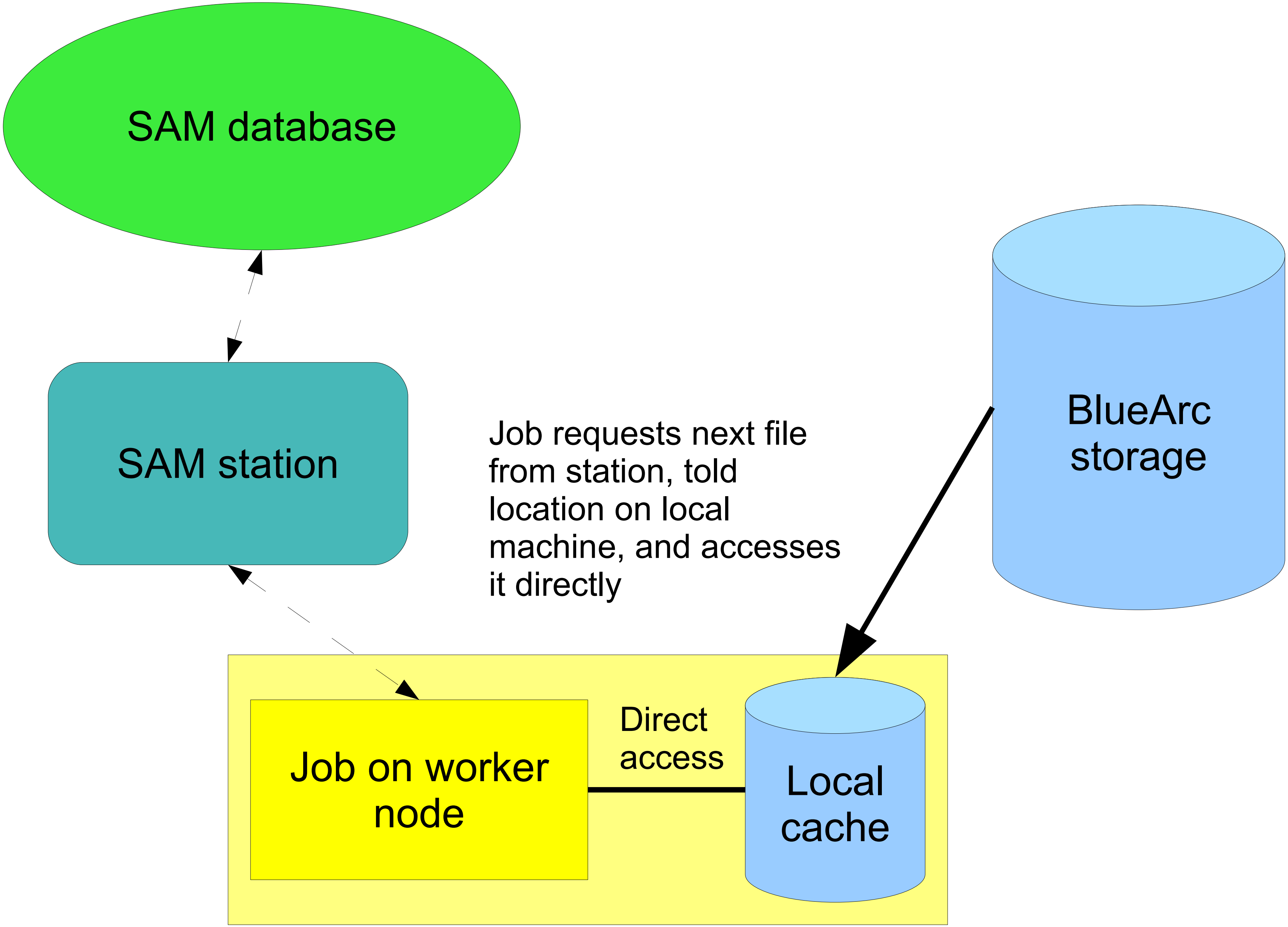


Job requests next file from station, told location, and copies it over

Usage scenarios - local

- SAM station with local cache disks
  - Add SAM managed cache on each worker node
  - Requires running SAM processes on each node
  - Requests not already in cache cause file to be copied over from BlueArc – station can limit simultaneous copies
  - Job given path to file in cache: can access it directly without local copy
  - Largely transparent to end users, but more complicated administration
  - Only a benefit if files in the cache get reused

Usage scenarios - local



SAM database

SAM station

Job requests next file from station, told location on local machine, and accesses it directly

BlueArc storage

Job on worker node

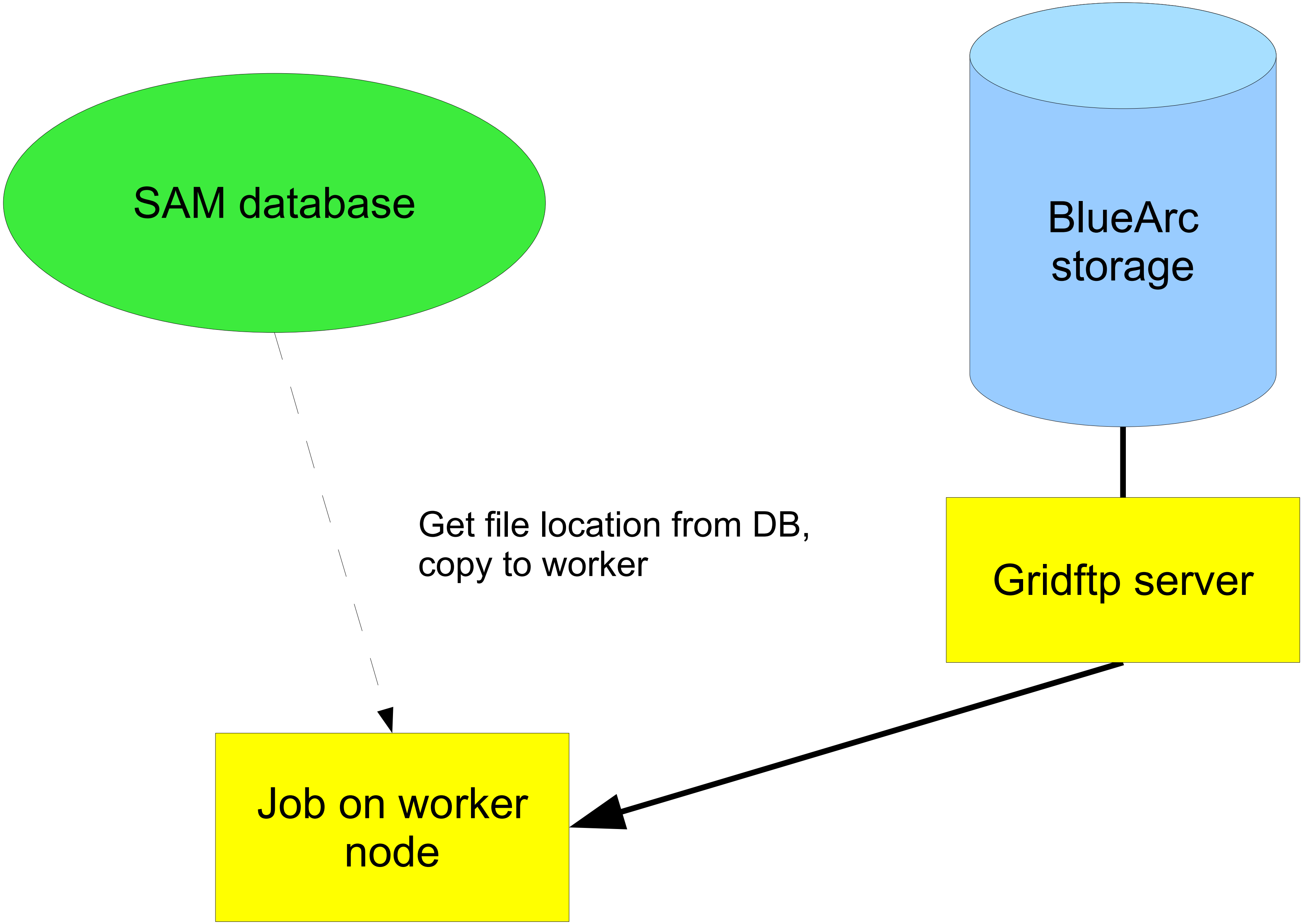
Direct access

Local cache

Usage scenarios - remote

- No SAM station
  - Job responsible for finding file location and copying it
    - if multiple locations would have to select appropriate one
  - Offsite replication (if required) would be entirely manual

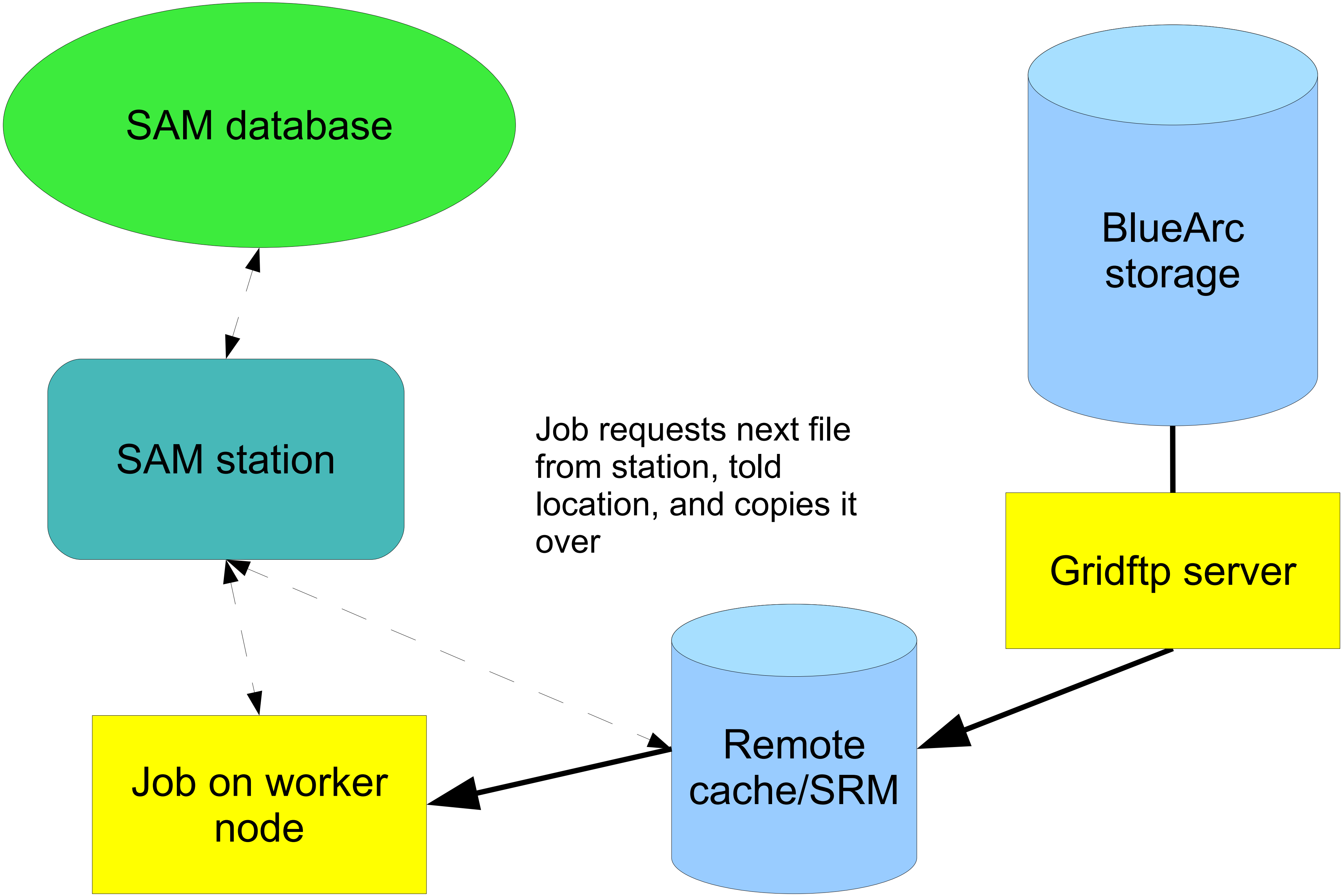
Usage scenarios - remote



Usage scenarios - remote

- With SAM station
  - Can configure station to manage offsite caches
    - plain disk cache requires running SAM services at remote site
    - SRM cache needs no extra remote services
  - Files copied to remote cache on demand
  - Need some method of deciding which remote cache a particular job should use

Usage scenarios - remote



Job requests next file from station, told location, and copies it over

Suggestions

- Various access methods not exclusive
- Important to get metadata right (painful to fix later)
- Concentrate on metadata & location for now
- Make cacheless station available for those who wish to use it