Argonne NATIONAL LABORATORY

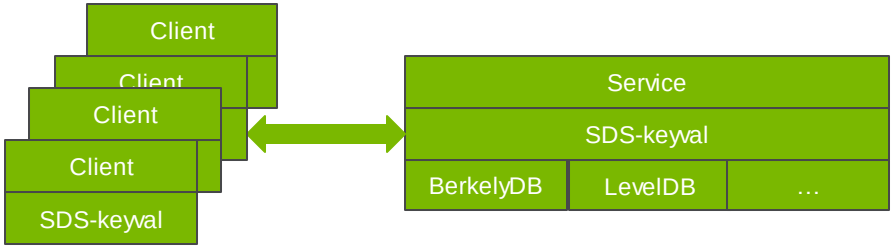
SDSKV: REMOTE KEYVAL SERVICE

ROB LATHAM

Mochi Bootcamp
8 February 2020

SDS-KEYVAL: MOCHI-PROJECTED DB

- Database vs file: same choice as in any other domain
  - Db: care more about accessing by ‘key’ than ‘offset
  - Db: key and value typically “small”
- Useful by itself
  - “Directory” application
- Also part of larger compositions
  - E.g. mobject: bulk data stored in bake regions, name-to-handle mapping stored in sds-keyval



SDS-keyval has several different DB backends.

SDS-KEYVAL: API CONCEPTS (PROVIDER)

- Provider configuration:
  - how to tie into margo, argobots, etc
  - Standalone provider will call margo_init: set up argobots, threads, mode
  - Will take margo identifier if a component
- Database configuration:
  - One or more databases hosted by provider
  - Type of database
  - Name
  - location

PROVIDER: C CODE

```c
#include <margo.h>
#include <sdskv-server.h>

int main(int argc, char **argv)
{
    /* ... */
    /* connect SDSKV to margo progess engine */
    sdskv_provider_t provider;
    sdskv_provider_register(mid, 1, SDSKV_ABT_POOL_DEFAULT, &provider);

    /* create a "Leveldb" database named "test-db" in the current directory,
      obliterating any prior instance */
    sdskv_database_id_t db_id;
    sdskv_config_t db_config = {
        .db_name = "test-db",
        .db_path = "",
        .db_type = KVDB_LEVELDB,
        .db_no_overwrite = 0
    };
    sdskv_provider_attach_database(provider, &db_config, &db_id);
    /* ... */
}
```

SDS-KEYVAL: API CONCEPTS (CLIENT)

- Initialize the library to obtain a "client" object
- Inform (e.g. config file or command line) client where to find server: result stored in a "provider handle"
- Open one (or more) databases on a given provider
- Now you can operate on the database
  - Put/get/erase
  - Get_multi/put_multi/erase_multi
  - List_keys/list_keys_with_prefix/list_keyvals
  - Key migration

CLIENT: C CODE

```c
#include <sdskv-client.h>

int main(int argc, char **argv)
{
    /* set up margo in usual way… */

    /* fire up the 'sdskv' client library to prepare to find a server */
    sdskv_client_t client;
    ret = sdskv_client_init(mid,  &client);

    /* resolve a mercury address to a 'handle' to the provider on a specific address */
    hg_addr_t server;
    margo_addr_lookup(mid, argv[1], &server);
    sdskv_provider_handle_t handle;
    ret = sdskv_provider_handle_create(client, server, 1, &handle);

    /* open one of the databases hosted by that provider */
    sdskv_database_id_t db_id;
    ret = sdskv_open(handle, "test-db", &db_id);

    int key=10; int value=99;
    ret = sdskv_put(handle, db_id, &key, sizeof(key), &value, sizeof(value));

    int get_value=0; size_t value_size=sizeof(get_value);

    ret = sdskv_get(handle, db_id, &key, sizeof(key), &get_value, &value_size);
    printf("key %d had value %d\n", key, get_value);

    /* omitted for space: cleanup */    return 0;
}
```

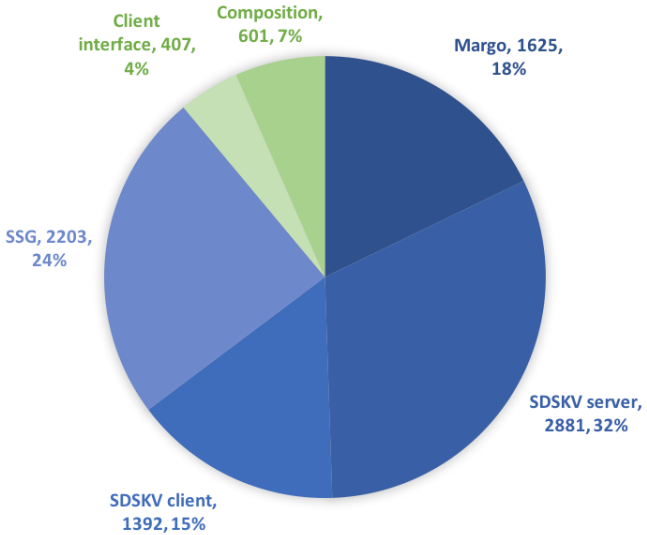
KEY MIGRATION

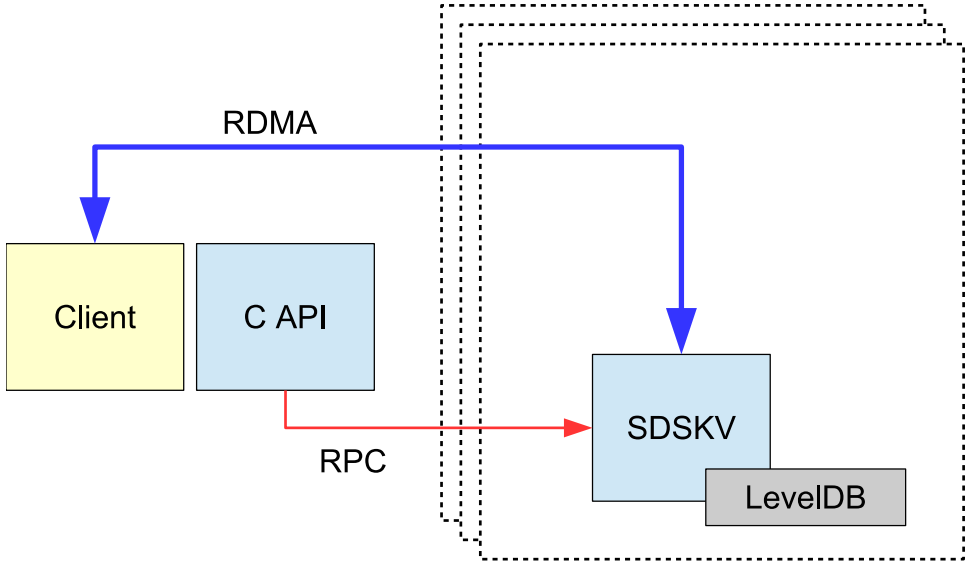
- Send one/some/all keys from one provider to a different provider
  - Client only sends rpc:
  - `sdskv_migrate_all_keys(kvphA, db_idA,`
    `sdskv_svr_addr_strB, mplex_idB, db_idB,`
    `SDSKV_REMOVE_ORIGINAL);`
- Remote end does the work
  - `margo_provider_forward()`: remote provider can itself become a client to another provider
  - "migrate entire database" uses the 'REMI' (REsource Migration) component
    - Packs up e.g. all leveldb files and transfers to new destination

EXERCISE

- Example (in slides and repo) puts an integer and gets it back
- Your task: store a bunch of strings instead of a single int
  - E.g. every word in /usr/share/dict/words
- Consult header files for API details:
  - https://xgitlab.cels.anl.gov/sds/sds-keyval/blob/master/include/sdskv-client.h
  - https://xgitlab.cels.anl.gov/sds/sds-keyval/blob/master/include/sdskv-client.hpp

SDSDKV: A DISTRIBUTED KEYVAL EXAMPLE





Dorier et al, "Methodology for the rapid development of scalable hpc data services", PDSW-DISC 2018

SDSDKV IN ACTION

```
// Determine personality type from global MPI ID.
sdsdkv_config_personality p = (
  (rank % 2 == 0) ? SDSDKV_PERSONALITY_SERVER: SDSDKV_PERSONALITY_CLIENT
);
// Define an SDSDKV instance configuration.
sdsdkv_config dkv_config = {
  MPI_COMM_WORLD, // Initializing MPI communicator
  p, // Process personality (client or server)
  rpc_thread_count, // RPC threading factor
  SDSDKV_HASHING_CH_PLACEMENT, // Hashing back-end
  SDSDKV_DB_LEVELDB,  // Database back-end type
  SDSDKV_COMPARE_DEFAULT, // K/V compare function
  "groupname", // Group identifier
  db_name, // Base path to database backing stores
  "ofi+tcp", // Communication protocol
};
sdsdkv_create(&dkvc, &dkv_config); // Create an SDSDKV instance named dkvc.
sdsdkv_open(dkvc); // Collectively open the dkvc instance.

// Client processes interact with key/value service
// while server processes field put/get requests.
sdsdkv_put(dkvc, &k, sizeof(k), &v, sizeof(v));
...
sdsdkv_get(dkvc, &k, sizeof(k), &v, &v_size);
...
sdsdkv_destroy(dkvc); // Collectively destroy SDSDKV instance.
```

SDSDKV SUMMARY

- Consider this a reference implementation
  - Emphasis on "reference"
  - Developed with specific application in mind
- Demonstrates compositing services
  - Ssg for group management
  - SDS-keyval for local db
  - Ch_placement to hash db keys to a server in group

ADDITIONAL FEATURES

- C++ interface
  - Abstracts some of the details and bookeeping
  - See `sdskv-client.hpp` for interface
  - https://xgitlab.cels.anl.gov/sds/sds-keyval/blob/master/include/sdskv-client.hpp
- Database management
  - Count, list databases
- "multiple item" interfaces
  - Able to batch up keys, values, lengths with '_multi' interface