ZHT:
A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table
Development tutorial

Tonglin Li, Xiaobing Zhou
Illinois Institute of Technology, Chicago, U.S.A
idea overview
architecture overview

Figure 3: ZHT architecture per node
file structure and readme

- See also `<FILE_STRUCTURE>`
- See also `<README>`
- [https://bitbucket.org/xiaobingo/iit.datasys.zh-t-mpi](https://bitbucket.org/xiaobingo/iit.datasys.zh-t-mpi)
Protocol abstraction

Diagram showing the relationships and methods involved in protocol abstraction.
How to choose Protocol

- zht.conf
  - PROTOCOL TCP
  - PROTOCOL UDP
  - PROTOCOL MPI
How to build ZHT

- Make executables for IP protocol family
  - make
  - Executables:
    - zht_cpptest/zht_ctest/zhtserver/c_zhtclient_lanl_threaded/c_zhtclient_threaded_test/cpp_zhtclient_threaded_test

- Make executables for MPI protocol family
  - make mpi
  - Executables:
    - zht-mpibroker
    - zht-mpiserver
How to launch ZHT servers

- For IP protocol family:
  - ./zhtserver -z zht.conf -n neighbor.conf
- For MPI protocol family:
  - mpiexec -np 4 ./zht-mpiserver -z zht.conf -n neighbor.mpi.conf
  - ./zht-mpibroker -z zht.conf -n neighbor.mpi.conf
- The ZHT client and YOUR_OWN_APP are not aware of the protocols
ZHT Language bindings

- C
- C++
- Recommendations
  - Always try the C++ binding since IT’S MORE CONVENIENT to pass user-defined composite data structure using OFFICIAL Google protocol buffer C++ binding
  - ZHT C binding depends on NON- OFFICIAL Google protocol buffer C binding, BUGS potently still
How to dev your ZHT apps

- Find C examples to call ZHT-client-API
  - See `<c_zhtclient_test.c>` for C example on how to call ZHT-client-API
  - See `<c_zhtclient_threaded_test.cpp>` and `<c_zhtclient_lanl_threaded.c>` for C example on how to call ZHT-client-API in multi-threaded context
Find C++ examples to call ZHT-client-API

- See <cpp_zhtclient_test.cpp> for C++ example on how to call ZHT-client-API
- See <cpp_zhtclient_threaded_test.cpp> for C++ example on how to call ZHT-client-API in multi-threaded context
How to dev your ZHT apps - walkthrough

- Assume the directory:
  - iit.datasys.zht-mpi
  - iit.datasys.zht-mpi/src
  - iit.datasys.zht-mpi/tutorial
  - iit.datasys.zht-mpi/tutorial/zhtsample.cpp
- See iit.datasys.zht-mpi/src/README to install < Google protocol buffers c binding, VERSION 0.15 > and < Google protocol buffers c++ binding, VERSION 2.4.1 >
- for example, you got iit.datasys.zht-mpi/tutorial/zhtsample.cpp as your app
- cd to iit.datasys.zht-mpi/src
- make or make mpi
- cd ../tutorial/
How to dev your ZHT apps - walkthrough

- `mkdir include` #create dir to hold ZHT header files your app may need
- `mkdir lib` #create dir to hold ZHT lib file your app needs to link to
- `cp ../src/*.h include/` #copy ZHT header files
- `cp ../src/libzht.a lib/` #copy ZHT lib file
- `vim comp.sh and enter`
  - `gcc -g -o zhtsample zhtsample.cpp -linclude/ -Llib/ -lzht -lstdc++ -lpthread -lprotobuf -lprotobuf-c`
How to dev your ZHT apps - walkthrough

- bash comp.sh #this will generate executable
  zhtsample
- cd to ../src, and start ZHT server as
  - ./zhtserver –z zht.conf –n neighbor.conf
- cd ../tutorial
- Run ZHT sample as
  - ./zhtsample –z ../src/zht.conf –n
    ../src/neighbor.conf
How to dev your ZHT apps – passing composite datastructure

- cd to iit.datasys.zht-mpi/tutorial/
- vim student.proto and enter
  - message Student {
    - required int32 id = 1;
    - required bool gender = 2;
    - required bytes firstname = 3;
    - required bytes lastname = 4;
    - required bytes address = 5;
    - required bytes phone = 6;
    - optional bytes hobbies = 7;
    - repeated bytes courses = 8;
  }
- protoc --cpp_out=. student.proto #this will generate student.pb.h and student.pb.cc
How to dev your ZHT apps – passing composite datastructure

- For example, you got iit.datasys.zht-mpi/tutorial/udtsample.cpp as app
- Other steps same as others in previous case
- vim comp.sh and append line as
  - gcc –g –o udtsample udtsample.cpp student.pb.cc -Iinclude/ -Llib/ -lzht -Istdc++ -lpthread -Iprotobuf -Iprotobuf-c
- launch zhtserver as mentioned before
- Run udtsample as
  - ./udtsample –z ../src/zht.con –n ../src/neighbor.conf
How to dev your ZHT apps – define your persistent storage

- When you launch zhtserver, it prompts:
  - Usage:
    - .*/zhtserver -z zht.conf -n neighbor.conf [-p port] [-f novoht_db_file] [-h(help)]
  - Using -f option, you specify the file to persist items you operated, e.g. novoht_db_file
How to dev your ZHT apps – run ZHT over MPI protocol(standalone mode)

- `vim iit.datasys.zht-mpi/src/zht.conf`, set
  - PROTOCOL MPI
- `make mpi`
- Launch 4 ZHT servers as
  - `mpiexec -np 4 ./zht-mpiserver -z zht.conf -n neighbor.mpi.conf : ./zht-mpibroker -z zht.conf -n neighbor.mpi.conf`
- Run your ZHT apps
How to dev your ZHT apps – run ZHT(cluster mode)

- see iit.datasys.zht-mpi/README
How to customize ZHT-augment client API

- Declare and define new C++ binding API in `<cpp_zhtclient.h>` and `<cpp_zhtclient.cpp>`,
  - Declare and define your operation code in `<Const.h>` and `<Const.cpp>`, e.g. `Const::ZSC_OPC_YOURS`, like `Const::ZSC_OPC_LOOKUP`
  - Learn from the existing API, e.g. `ZHTClient::lookup`, the stack looks like:
    - `int ZHTClient::lookup(const char *key, char *result)`
    - `int ZHTClient::lookup(const string &key, string &result)`
    - `int ZHTClient::commonOp(const string &opcode, const string &key, const string &val, const string &val2, string &result, int lease)`
    - `string ZHTClient::commonOpInternal(const string &opcode, const string &key, const string &val, const string &val2, string &result, int lease)`
- Declare and define delegation for new API in `<c_zhtclientStd.h>` and `<c_zhtclientStd.cpp>`
How to customize ZHT-augment client API

- Declare and define C binding for new API in `<c_zhtclient.h>` and `<c_zhtclient.cpp>`
How to customize ZHT-augment server implementation

- Declare and define server implementation for new client API in `<HTWorker.h>` and `<HTWorker.cpp>`
- Learn from existing impl, e.g. lookup, the stack looks like:
  - string HTWorker::run(const char *buf)
    - Be sure to add the operation dispatch code like:
      - if (zpack.opcode() == Const::ZSC_OPC_YOURS)
  - string HTWorker::lookup_shared(const ZPack &zpack)
How to dev your ZHT apps

- demo
Questions?

Tonglin Li, Xiaobing Zhou
tli13@hawk.iit.edu, xzhou40@hawk.iit.edu
http://datasys.cs.iit.edu/projects/ZHT/