Ciências ULisboa | Informática

Apache ZooKeeper

Pedro Ferreira / Bernardo Ferreira

Departamento de Informática

Faculdade de Ciências da Universidade de Lisboa

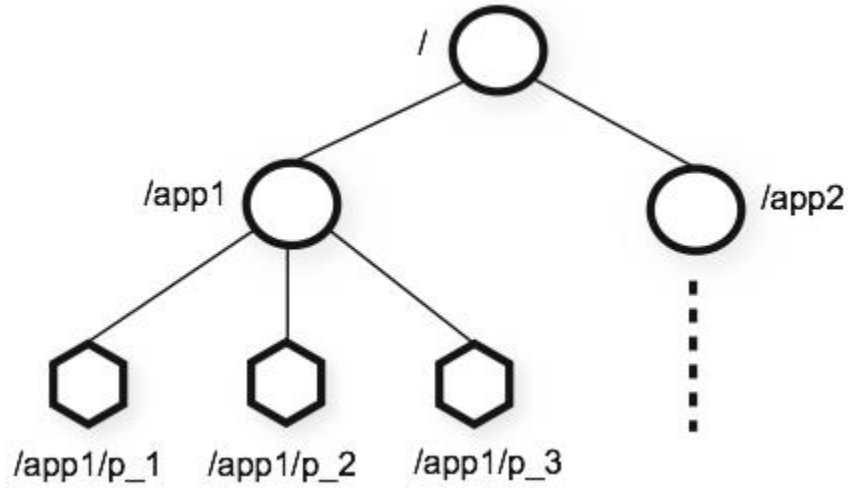
Fonte da imagem: http://www.dsn.jhu.edu

Apache ZooKeeper

- Serviço para Coordenação de Sistemas Distribuídos
  - Fiável, escalável e de alta disponibilidade
  - Não serve para guardar dados, mas sim meta-dados
    - » Ex: IPs, timestamps, versões, etc.
  - Usado por outros serviços, como:
    - » Base dados replicadas, sistemas de sincronização, Blockchain, etc.
- Multi-cliente
  - pode ser usado por múltiplos clientes simultaneamente
- Bibliotecas e APIs em múltiplas linguagens
  - C, Java, Perl, Python
  - Command Line Interface (CLI) também disponível

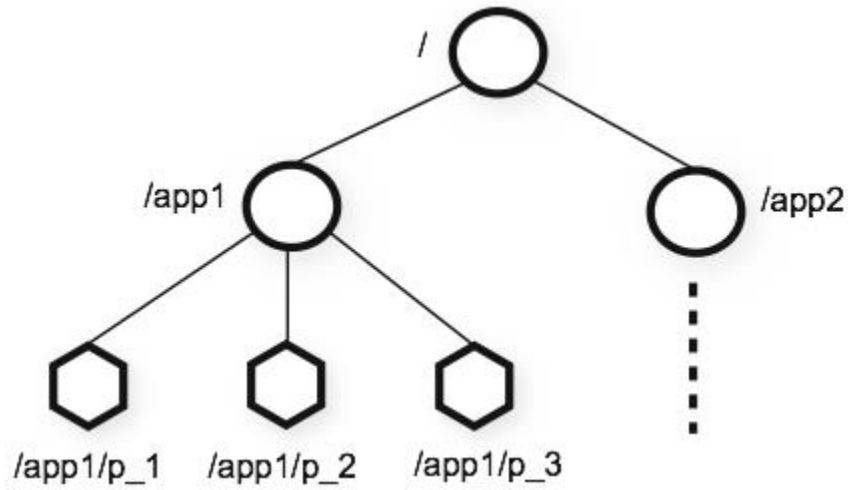
Modelo de Dados ZooKeeper

- **Como é que os metadados escritos pelos clientes podem ser guardados e organizados dentro do ZooKeeper?**



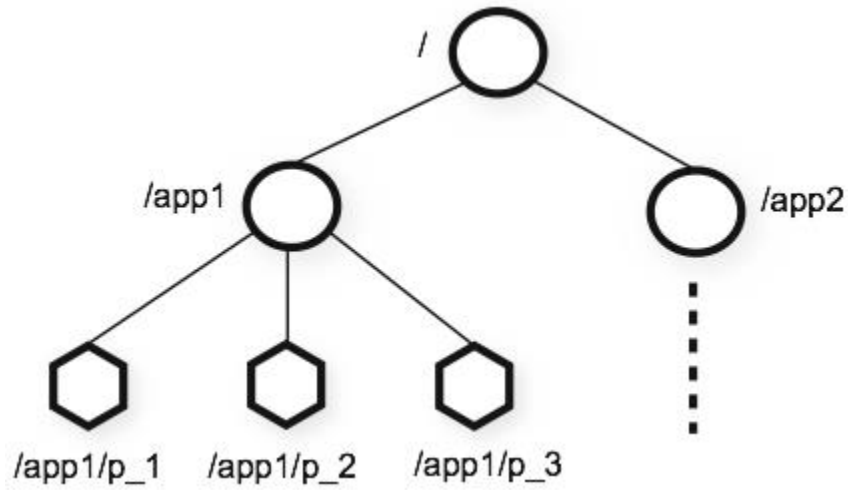
Modelo de Dados ZooKeeper

- **Znode**
  - Guarda metadados em memória (max 1 MB por znode)
  - Organização hierárquica, i.e. um nó pode ter vários “filhos”
  - Nomes usam notação estilo sistema de ficheiros UNIX. Ex: /app1/p_1/...
- **Tipos de Znode**
  - **Normal**
  - **Efémero**: não pode ter filhos e só existe enquanto sessão cliente que o criou existir
  - **Sequencial**: é atribuído automaticamente ao nó um numero de sequencia
    - » Ex: /app1/p_0000000001
    - » Tanto nós normais como efémeros podem ser simultaneamente sequenciais



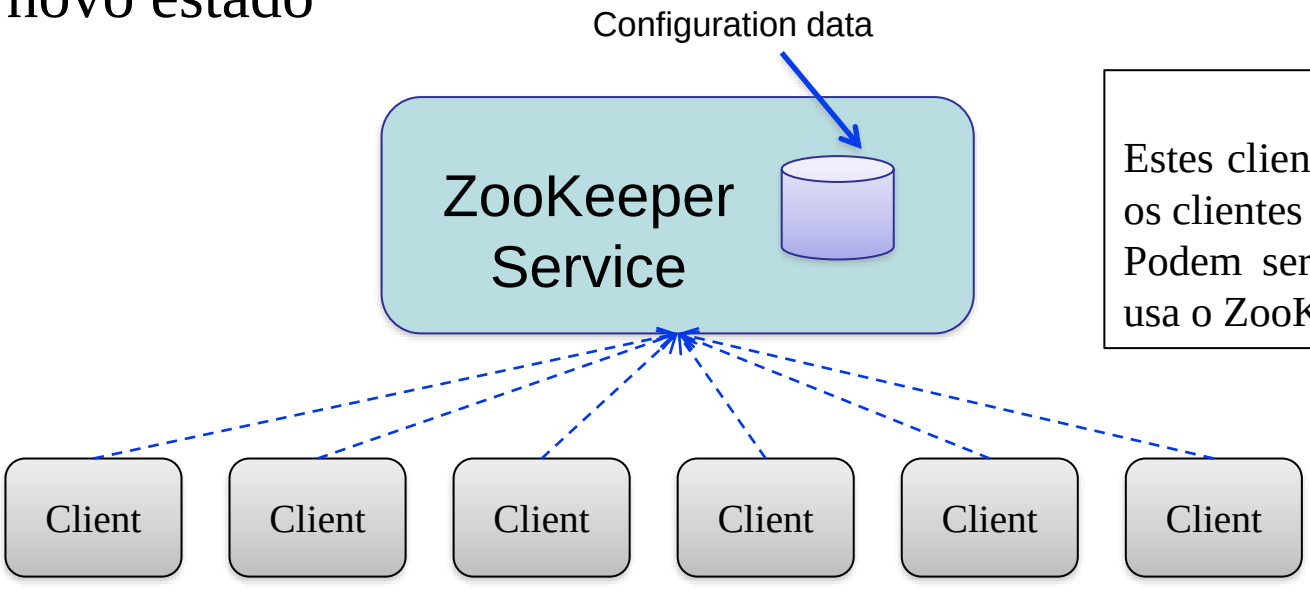
Modelo de Dados ZooKeeper

- **Mecanismo de *Watch***
  - Permite notificar clientes quando há alterações num nó ou nos seus filhos
  - Ex: *zoo_wget_children* (... , ”/app1”, ...)
    - » Método invocado quando se quer ser notificado da adição ou remoção de filhos ao nó “/app1”



ZooKeeper - Exemplos de Utilização

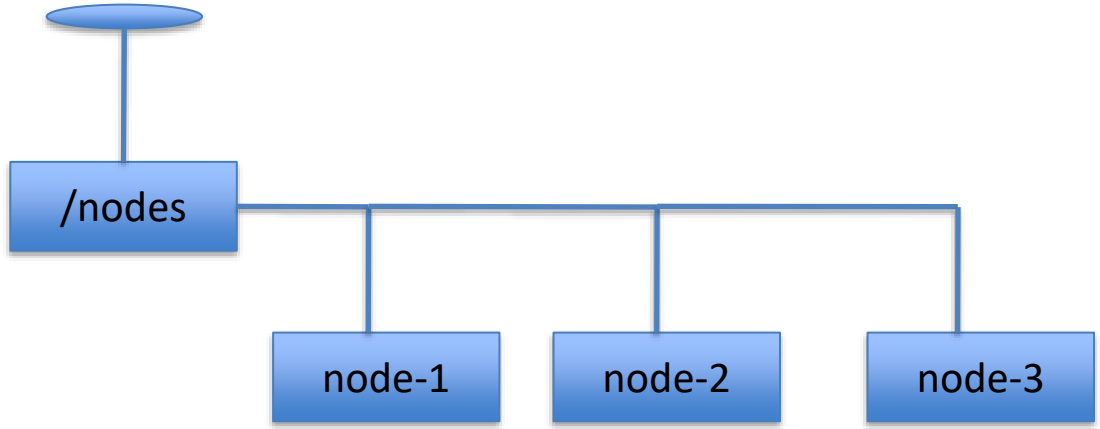
- Gestão de Configuração Distribuída
  - Um cliente cria um Znode **c** para guardar configuração de um SD
  - Outros clientes fazem *watch* do Znode **c**
  - Quando há um update a **c**, clientes são notificados e verificam novo estado



Nota:
Estes clientes não são necessariamente os clientes finais da aplicação.
Podem ser servidores de um SD que usa o ZooKeeper para coordenação.

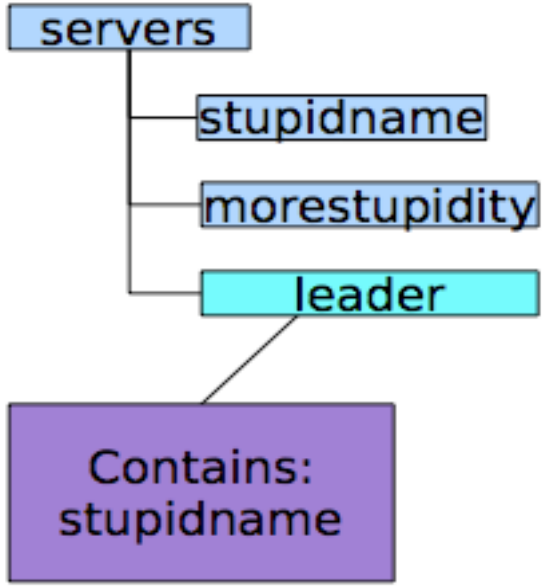
ZooKeeper - Exemplos de Utilização

- Gestão de membros de grupo
  - Cliente que gere o grupo cria um Znode **g**
  - Cada um dos restantes clientes criam um Znode efémero filho de **g**
  - Para receber atualizações sobre o grupo, clientes fazem *watch* a **g** e vêm os filhos



ZooKeeper - Exemplos de Utilização

- Eleição de líder
  - Verificar quem é líder
    - » getData (... , "/servers/leader", ...)
  - Se houver um líder, seguir o líder
  - Se não houver líder, tentar registar como líder
    - » create (... , "/servers/leader", ZOO_EPHEMERAL, ...)
  - Se não conseguir seguir nem criar, voltar ao primeiro passo



API ZooKeeper

- Cada método tem duas versões: síncrono e assíncrono
  - Métodos síncronos não bloqueiam, definindo antes uma função que é invocada quando chega resposta do servidor
- Métodos síncronos/assíncronos
  - *create (path, data, acl, flags)*
  - *delete (path, version)*
  - *exist (path, watch)*
  - *getData (path, watch)*
  - *setData (path, data, version)*
  - *getChildren (path, watch)*
- Métodos apenas assíncronos
  - *sync (path)*
- Todos os métodos estão definidos em ficheiro **zookeeper.h**

API ZooKeeper em C – Síncrona

- *zhandle_t *zh*
  - Token de sessão para comunicação com o zookeeper

- *zookeeper_init (...)*
  - Função que inicia comunicação com zookeeper e devolve token de sessão

```
ZOOAPI zhandle_t*
zookeeper_init (
  const char *host,        /* comma separated host:port pairs */
  watcher_fn  fn,          /* watcher callback function */
  int recv_timeout,        /* session expiration time */
  const clientid_t *clientid, /* session id for reconnection */
  void *context,   /* context of the handle object */
  int  flags       /* for future use, should be set to zero*/
)
```

API ZooKeeper em C – Síncrona

- *int zoo_create (...)*
  - Cria um novo nó no caminho indicado por *path*, com valor *value*

```
ZOOAPI int zoo_create (
   zhandle_t *zh,                    /* the zookeeper handle */
   const char *path,                 /* the path of the node */
   const char *value,                /* the data to store in the node, or NULL*/
   int valuelen,                     /* the data size, or -1 if its NULL*/
   const struct ACL_vector *acl,     /* ACL of the node, must not be NULL */
   int flags,                       /* creation flags (ZOO_EPHEMERAL, ZOO_SEQUENCE,
                                        a bitwise OR of the two, or 0)*/
   char *path_buffer,                /* used to store the name assigned to the
                                        node, if flags has ZOO_SEQUENCE */
   int path_buffer_len )             /* used to store path_buffer length*/
```

API ZooKeeper em C – Síncrona

- *int zoo_get_children (zhandle_t *zh, const char *path, int watch, struct String_vector *strings)*
  - Escreve em *strings* os nós listados em *path e* devolve código de erro

- *int zoo_get (zhandle_t *zh, const char *path, int watch, char *buffer, int* buffer_len, struct Stat *stat)*
  - Copia para *buffer* os dados guardados no nó indicado por *path*

- *int zoo_set (zhandle_t *zh, const char *path, const char *buffer, int buflen, int version)*
  - Escreve no nó indicado por *path* os dados guardados em *buffer*

API ZooKeeper em C – Síncrona

- *int zoo_exists (zhandle_t *zh, const char *path, int watch, struct Stat *stat)*
  - Verifica se o nó indicado por *path* existe, devolvendo a resposta (código ZOK, ZNONODE, ou outro)

- *int zoo_delete (zhandle_t *zh, const char *path, int version)*
  - Apaga o nó indicado por *path*

- *int zookeeper_close (zhandle_t *zh)*
  - Função que termina uma sessão e devolve código de erro

Exemplo

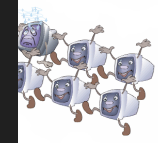
- Ficheiro Zoo.c no Moodle

```c
/* Main Function */
int main(int argc, char *argv[]) {
    int i, retval;
    const char* host_port = "localhost:2181";
    const char* zoo_root = "/";
    zoo_string* children_list = (zoo_string *) malloc(sizeof(zoo_string));

    /* Connect to ZooKeeper server */
    zh = zookeeper_init(host_port, my_watcher_func, 2000, 0, NULL, 0);
    if (zh == NULL) {
        fprintf(stderr, "Error connecting to ZooKeeper server!\n");
        exit(EXIT_FAILURE);
    }

    /* Get the list of children synchronously */
    retval = zoo_get_children(zh, zoo_root, 0, children_list);
    if (retval != ZOK)  {
        fprintf(stderr, "Error retrieving znode from path %s!\n", zoo_root);
        exit(EXIT_FAILURE);
    }
    fprintf(stderr, "\n=== znode listing === [ %s ]", zoo_root);
    for (i = 0; i < children_list->count; i++)  {
        fprintf(stderr, "\n(%d): %s", i+1, children_list->data[i]);
    }
    fprintf(stderr, "\n=== done ===\n");

    /* Finally close the ZooKeeper handle */
    zookeeper_close(zh);
    return 0;
}
```



Exemplo

- Ficheiro Zoo.c no Moodle
  - Compilar com:
    - » gcc zoo.c -o zoo -lzookeeper_mt
  - Usar CLI para adicionar alguns nós e correr ./zoo para ver alterações
    - » /usr/lib/zookeeper/bin/cli_mt 127.0.0.1:2181
      - create /node1
      - create /node2
      - delete /node1
      - create /node3

Instruções para laboratórios. PCs pessoais têm que instalar zookeeper e podem ter que usar outros caminhos (-I ... –L ...)

API ZooKeeper em C – Watchers

- Funções em que é possível definir *watcher* levam um **w** antes do nome da função e juntam mais dois argumentos. Ex:
  - Ex: *int zoo_wget_children (zhandle_t *zh, const char *path, watcher_fn watcher, void* watcherCtx, struct String_vector *strings)*
    - » Função leva *w* antes do nome
    - » *watcherCtx* dados do utilizador a passar à função
    - » *watcher* é o nome da função a invocar qnd houver updates:

```
typedef void
(* watcher_fn)(
  zhandle_t *zh, /* the zookeeper handle */
  int type,/* event type, e.g. ZOO_SESSION_EVENT */
  int state,/* connection state, e.g. ZOO_CONNECTED_STATE */
  const char *path, /* znode path for which the watcher is
  triggered. NULL for session events */
  void *watcherCtx/* watcher context object */
)
```

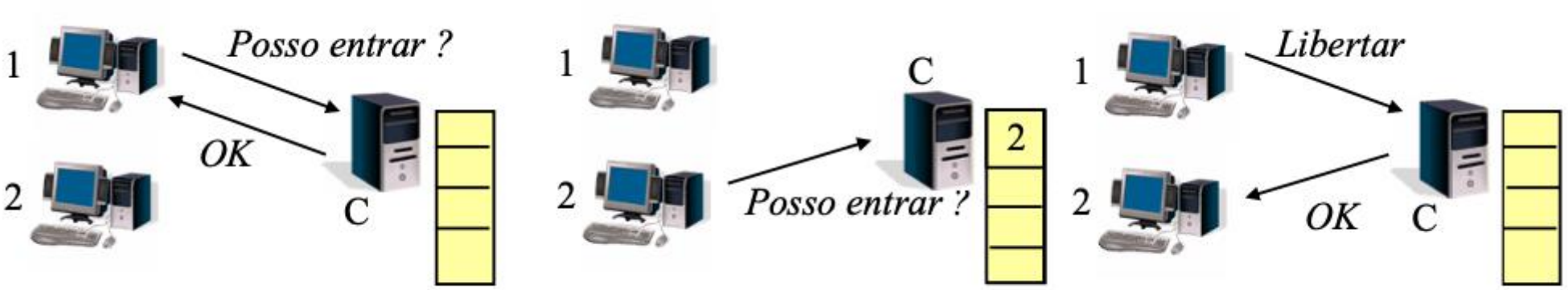
Exemplos

- Ver ficheiros zoochildmaker.c e zoochildwatcher.c no Moodle
  - Exemplo com um znode normal "/children"
  - zoochildmaker adiciona periodicamente nós efémeros sequenciais a "/children" e depois termina
  - zoochildwatcher faz *watch* aos filhos de "/children" e imprime lista sempre que há alterações

- Ver ficheiros zoodataupdater.c e zoodatawatcher.c no Moodle
  - Exemplo com um znode normal "/MyData"
  - zoodataupdater periodicamente altera os dados de "/MyData"
  - zoodatawatcher faz *watch* a alterações em "/MyData" e imprime novos dados

Exercicio

- Implementar um serviço de exclusão mútua centralizado baseado no ZooKeeper
  - Ver aulas téoricas sobre exclusão mútua



API ZooKeeper em C

- Mais informações:
  - Ver ficheiro zookeeper.h
    - » Define todas as funções, flags e códigos de erro possíveis
  - ZooKeeper Programmer's Guide
    - » https://zookeeper.apache.org/doc/r3.3.6/zookeeperProgrammers.html#ZooKeeper+C+client+API
  - ZooKeeper Book
    - » https://www.oreilly.com/library/view/zookeeper/9781449361297/
  - Apache ZooKeeper Essentials Book
    - » https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781784391324