Documentação Oficial do Anahy 3

Gerson Geraldo H. Cavalheiro, Cícero Augusto de S. Camargo, Alan S. de Araújo

[1]Programa de Pós-Graduação em Computação (PPGC)
Universidade Federal de Pelotas (UFPEL)

{cadscamargo,asdaraujo,gerson.cavalheiro}@inf.ufpel.edu.br

1. Introdução

2. Modelo de execução

3. Código Fonte

O código fonte de Anahy 3 é escrito em C++, usando uma modelagem orientada a objetos e threads POSIX para a execução de múltiplos threads no nível de sistema.

3.1. Classes

Job

JobId

JobAttributes

VirtualProcessor

Daemon

SchedulingOperation

AnahyVM

4. Biblioteca athread.h

A biblioteca **athread.h** oferece uma programação a alto nível para lançar atividades concorrentes, similar a interface do padrão POSIX Thread. A interface fornece um modelo *fork/join* para descrever programas em termos de *threads*. Uma camada intermediária entre **athread.h** e o núcleo do programa é responsável por identificar a concorrência fornecida a partir da infertace em pequenas partes, chamadas tarefas e implementadas pela classe *Job*. A partir disso é possível construir um DCG para representar as tarefas do programa. Esta camada intermediária é implementada na classe *AnahyVM*.

aInit

Este método coleta dados fornecidos pelo programador para configurar o ambiente de execução. A_INIT tem a seguinte definição: `void aInit(int argc, char** argv)`, os dados passados por linha de comando são: número de processadores virtuais, modo de execução do ambiente, modelo de escalonamento das tarefas e mais alguns atributos que serão apresentados adiante.

athread_create

O A_THREAD_CREATE lança as atividaes concorrentes ao ambiente de execução. Este método fornece a seguinte definição: `int athread_create(athread_t* thid, athread_attr_t* attr, pfunc function, void args)`. Novos *threads* são criados para executarem a função em `pfunc function` que os descreve. Os dados de entrada para função são especificados em `args`. O parâmetro `thid` é usado para identificar o novo *thread* criado. O argumento `attr` especifica os atributos dos *threads*, sendo estes, custos de computação, número de *joins* sofridos entre outros que serão vistos na seção Atributos.

athread_join

A_THREAD_JOIN fornece a definição: `int athread_join(athread_t thid, void** result)`. Cada *thread* identificada por `thid` realiza uma operação de sincronização e `result` guarda a área de memória com o resultado final do *thread*, esse dado é definido pela primitiva ATHREAD_EXIT.

athread_exit

Este método cuja definição é `void athread_exit(void* value_ptr)` é chamado ao final de cada função que define um *thread*, onde `value_ptr` guarda o resultado final da execução em uma área de memória própria da tarefa.

aTerminate

A_TERMINATE é definida por `void aTerminate()` e é chamada no final da execução do programa para liberar a área de memória utilizada pelo ambiente. Ao término da execução do programa a função redefinir possíveis atributos do sistema que foram alterados durante a execução da aplicação.

5. Conclusão

Anahy é bom demais!