Sistema de control de evaluaciones semi-paralelas

Juan Francisco Cardona Mc'Cormick

4 de Mayo de 2017

1. Introducción

Una de las características de los procesadores y sistemas operativos modernos es la capacidad de ejecutar más de una tarea simultáneamente. En esta práctica, vamos a tomar dicha característica para implementar un sistema de ejecución de expresiones seudo-paralelo.

Para llevar a cabo dicho sistema se propone una arquitectura, dividida en tres capas. Cada capa tiene una función descrita y definida en la siguientes secciones.

El objetivo de la práctica, es implementar la arquitectura propuesta a través de los conceptos y herramientas vistos hasta el momento en clase.

2. Arquitectura propuesta

En la figura 1 se observa la arquitectura del sistema a implementar. Esta arquitectura se asemeja a un árbol, donde los nodos son los procesos y los arcos son mecanismo de intercambio entre los procesos.

Se distiguen tres tipos de procesos (*nodos*):

- **Proceso control del sistema** (`ctrlsis`): También llamado proceso de nivel 0. Se encarga ejecutar una serie de procesos definidos por el fichero de descripción (`ctrsis.cfg`). Este es responsable de pasar los mensajes de contexto entre los procesos de nivel 1 (`ctreval`), este mensaje es pasado en el orden que aparecen en el fichero de descripción (`ctrsis.cfg`).

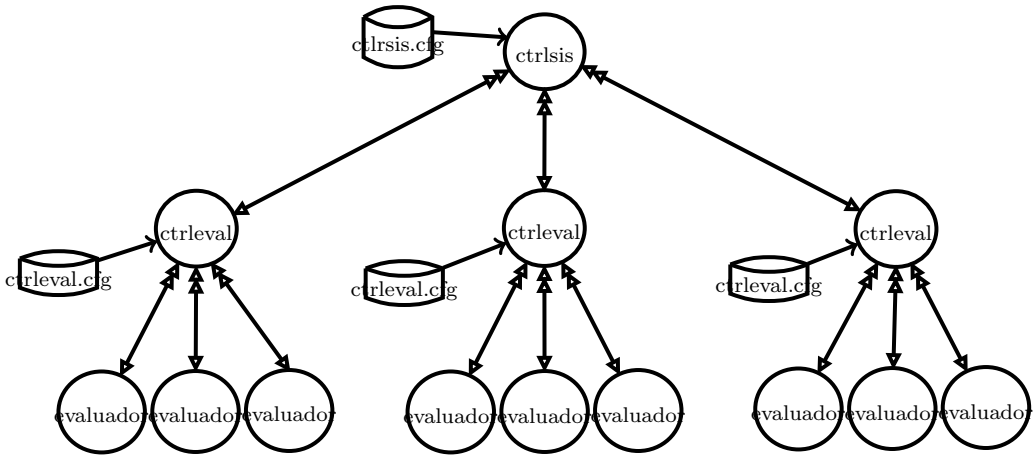

Figura 1: Modelo del sistema de control

- **Proceso de control de evaluaciones** (`ctrleval`): También llamado proceso de nivel 1. Este proceso se encarga de controlar un conjunto de procesos de nivel 2 (`evaluador`) que se encargarán de realizar las evaluaciones de cada expresión descrita por un fichero de configuración ubicado en un directorio en particular (`ctrleval.cfg`). El proceso de control se encarga de obtener las variables de ambiente del proceso padre de nivel 0 y pasarla a cada uno de sus procesos hijos (nivel 2) en el orden que estos están definidos en fichero de configuración.

- **Proceso evaluador** (`evaluador`): El `evaluador` se encarga de recibir una expresión al inicio del programa y cada vez que llega por su entrada estándar un entorno de variables, estas son ejecutadas por el y retornar su respuesta por la salida estándar.

Existe un tipo de conexión entre los procesos: *tipo 1*.

En la conexión tipo 1 - 2 se permite que el proceso que está en el nivel inferior reciba información por la entrada estándar y retorne información por la salida estándar al proceso que está conectado en la parte superior de la conexión. También se envia mensajes de error por el error estándar.

Figura 2: Conexión tipo 1

2.1. Componentes

2.1.1. Fichero de configuración `ctrlsis.cfg`

Este fichero de configuración tiene la información referente a toda la jerarquía de procesos que va a tener el sistema. El fichero de configuración tiene la siguiente sintaxis:

```
ArchCfg    →   Control+
Control    →  'Control' ID '{' Path EvalCfg  '}'
Path       →  (('.'|'..')'/')+ID('/'ID)*
ID         →  [a..zA..Z]([a..zA..Z]|[0..9])*
EvalCfg    →  ID'.'cfg
```

Un ejemplo de un fichero de configuración:

```
Control Uno { ./nivel1/hijo1 hijo1.cfg }
Control Dos { ./nivel1/hijo2 hijo2.cfg }
```

Este muestra dos procesos de nivel 1 llamados `hijo1` e `hijo2`; y sus respectivos ficheros de configuración `hijo1.cfg` e `hijo2.cfg`.

2.1.2. Fichero de configuración de evaluador `ctrleval.cfg`

Este fichero de configuración tiene la información de las expresiones expresiones que se van a evaluar en cada ejecución secuencial de los procesos evaluadores. La siguiente es la sintaxis:

```
EvaluadorCfg →  Evaluador+
Evaluador    →  'Evaluador' NUMERO '{' Asignaciones '}'
Asignaciones →  Asignación+
Asignación   →  ID '=' Expresión ';'
Expresión    →  Termino (('+'|'-') Termino)*
Termino      →  Factor (('*'|'/') Factor)*
Factor       →  NUMERO
```

```
              |   ID
              |   '(' Expresión ')'
ID            →   [a..zA..Z]([a..zA..Z]|[0..9])*
NUMERO        →   ([1..9][0..9]* | '0')
```

Un ejemplo de un fichero de configuración:

```
Evaluador 1 {
a = 10 + b;
b = 30 * c;
c = 12 / a;
d = a + b - (c + d);
}
Evaluador 2 {
a = 12;
b = a + b;
d = e - d;
e = a * b;
}
```

Aquí se muestra dos procesos de nivel 2 que deben ser lanzados y cada una de las expresiones que debe evaluar.

2.1.3. Control del sistema *ctrlsis*

Este proceso es el encargado de crear la jerarquía de procesos, a pasar mensajes entre los procesos de nivel 1 y verificar el estado del procesos de nivel 1 creados por el. Cuando el proceso de control de sistema se inicia, este lee fichero de configuración (2.1.1) y a partir de la descripción allí obtenida construye la jerarquía de procesos.

Cada control recibe la información de la ubicación y nombre del fichero de configuración a través de dos variables de ambiente:

```
DIRDETRABAJO=./nivel1/hijo1
FICHEROCFG=hijo1.cfg
```

Una vez creada la jerarquía, el proceso `ctrlsis` realiza la siguientes labores.

- monitoreo de los procesos de nivel 1,

- pasar el entorno entre los procesos de nivel 1,
- comunicarse con el usuario final para controlar la ejecución.

Paso de entorno: A cada `ctreval` se le da un turno basado en un posición en el fichero de configuración para en cada `evaluador` un conjunto de expresiones. El turno se inicia cuando el proceso de control del sistema pasa el entorno actual al primer proceso de `ctreval` pasando el entorno a través de la entrada estándar de este (`ctrleval`); este `ctrleval` pasa a cada proceso `evaluador` el entorno. La primera vez que se pasa el entorno al primer `ctrleval`, el entorno está vacío. Cuando el proceso `ctrleval` termina de procesar en entorno en sus `evaluadores`, el entorno recolecador pasa al siguiente `ctrleval` hasta que llege al último, el entorno modificado es mostrado en la salida estándar del `ctrleval` y un nueva ronda comienza con el entorno obtenido en la anterior.

La sintaxis del entorno es la siguiente:

```
Entorno   →  '{' Variable* '}+'
Variable →  ID '=' NUMBER ';'
ID           →  [a..zA..Z]([a..zA..Z]|[0..9])*
NUMBER       →  [-]([1..9][0..9]* | '0')
```

- Un entorno vacío:

  { }

- Un entorno con valores

  `{ a = 10; b = 20; c = -20; }`

Monitoreo de los `ctrleval` Los `ctrleval` y los procesos que controlan `evaluadores` pueden fallar por cualquier circunstancia: divisiones por cero, no existe le directorio, el fichero de configuración del evaluador corrupto, etc. El proceso de control del sistema debe estar pendiente de los mensajes de error que retorna cada `ctrleval` en cada momento o de la posible terminación de cada uno de ellos. Este monitoreo no se hace por ronda, sino que debe ser **permanente**. Todos los errores son recibidos de cada proceso `ctrleval` por el error estándar y sacados por el error estándar del `ctrlsis`.

La sintáxis de los mensajes de error:

```
FormatoError →  Error*
Error        →  'Control' ID '{' 'Evaluador' NUMERO ':' NUMERO '}'
```

Donde el `ID` es el identificador del `ctrleval`, el primero `NUMERO` es el número del evaluador y el segundo `NUMERO` es el número del error tipificado. Estos errores son propios de cada aplicación y deben estar descritos en el `Readme` del proyecto.

```
Control Uno  Evaluador 1 : 1
Control Dos  Evaluador 3 : 10
```

Comunicación usuario final: El proceso `ctrlsis` también recibe por la entrada estándar de este, mensajes de control del usuario final. La siguiente es la sintáxis del lenguaje del usuario final:

```
UsuarioFinal → Accion
Accion       → 'mostrar'
             | 'ocultar'
             | 'parar'
```

Donde la acción `mostrar` muestra los mensajes del entorno al final de cada ronda; la acción `ocultar` oculta los mensajes de entorno al final cada ronda; y la acción `parar` para la ejecución del `ctrlsis` y de toda la jerarquía de procesos. Por omisión el sistema comienza con la acción de `mostrar`.

2.1.4. Proceso de control de evaluaciones (`ctrleval`)

Una vez iniciado este proceso, este procede a ubicarse en el directorio determinado por la variable de ambiente `DIRDETRABAJO` y a buscar allí el fichero de configuración, también determinado por la variable de ambiente `FICHEROCFG`.

El fichero de configuración del evaluador 2.1.2 contiene varios conjuntos de expresiones que este debe computar. Cada conjunto es evaluado en un turno, cuando termina los conjuntos se vuelve a comenzar.

Un turno inicia cuando el proceso de control del sistema envia un entorno a través de la entrada estándar. Este se encarga de pasar al primer evaluador

por el orden del número de identificación (`NUMERO`). Cada `evaluador` procesa los mensajes y ejecuta cada asignación que cambia el entorno hasta que las asignaciones han terminado y un nuevo entorno se ha generado se pasa al siguiente proceso `evaluador`. Cuando el último proceso evaluador proceso su enterno, este nuevo entorno es retornado al proceso de control evaluador (`ctrlsis`).

2.1.5. Proceso evaluador (`evaluador`)

El proceso evaluador obtiene la información sobre las expresiones a evaluar a través de la línea de comandos, así:

```
evaluador <ctrleval.cfg> <numero evaluador>
```

Donde `evaluador` es el nombre del ejecutable del evaluador, el primer argumento (`ctrleval.cfg`) es el nombre del fichero de configuración donde se encuentra la descripción de la configuracion; y el segundo argumento (`numero evaluador`) es el número del evaluador dentro del fichero de configuración.

El proceso `evaluador` recibe a través de la entrada estándar el entorno. Lo evalua y lo envia a la salida estándar. El proceso no termina y no que espera por nuevos entornos hasta que la salida estándar se cierre.

3. Requerimientos administrativos

3.1. Grupo

1. Grupos: máximo tres estudiantes.

2. Cada grupo debe escoger un nombre que identifique al grupo, este nombre utiliza la misma sintaxis que los identificadores en Java.

3. Deben crear un proyecto privado en el manejador de repositorios `https://bitbucket.org` con el mismo nombre del grupo escogido e invitando al profesor y al monitor a participar del proyecto (no pueden tener el rol de administrador o dueño).

4. Invitar al profesor (Usuario: `fcardona@eafit.edu.co`) y el monitor (Usuario: en) en *bitbucket*.

3.2. Lenguajes de programación

Los lenguajes aceptados para la práctica son C o C++. Se puede llamar funciones de C desde C++, pero en el sentido inverso, al hacerlo recuerde identificar correctamente el espacio de nombres de las funciones de C.

3.3. Sistemas operativos

La práctica debe ser realizada en Linux (Ubuntu a partir de la distributicón 17.04).

3.4. Estructura del repositorio

La siguiente es la estructura de los directorios que tener el respositorio para manejar el proyecto.

```
+
|= - miembros.xml
|  + src
|  - Makefile
|  - Readme
|  - Install
```

`miembros.xml` fichero que contiene la definición de los miembros del grupo. El siguiente es el contenido de un posible grupo.

```
<grupo>
   <nombre>Diosnosllevedesumano</nombre>
   <repositorio>https://bitbucket.org/Diosnosllevedesumano</repositorio>
   <miembro>
      <nombre>Juan Francisco Cardona McCormick</nombre>
      <codigo>198610250010</codigo>
      <grupo>032</grupo>
      <email>fcardona@eafit.edu.co</email>
   </miembro>
   <miembro>
      <nombre>Juan Pablo Gaviria Salazar</nombre>
      <codigo>20140039010</codigo>
      <grupo>032</grupo>
      <email>jgaviri6@eafit.edu.co</email>
   </miembro>
</grupo>
```

3.5. Otro software y bibliotecas de funciones

En el fichero `Readme` debe indicar las bibliotecas o el software adicional que han utilizado para compilar sus programas.

3.6. Primera entrega

8 de Junio a las 8:00 a.m. hora oficial de Colombia. En ese mismo instante un procedimiento automático bajara los repositorios correspondientes. Deben actualizar sus repositorios antes. Grupo que no haya invitado al profesor no será tenido en cuenta.