Progetto di ingegneria Informatica: programmazione di sistemi eterogenei paralleli

Niccolò Nicolosi

16/06/2022

1 Motivazioni e obiettivi

Il progetto europeo The European PILOT, avviato in data 1 dicembre 2021 e coordinato dal Barcelona Supercomputing Center (BSC), mira alla realizzazione di una macchina di supercalcolo i cui acceleratori siano interamente basati sull'architettura open source RISC-V e sue estensioni. Tale macchina necessita di diverse componenti, sia hardware che software, per poter sfruttare risorse tecnologiche eterogenee, compresi gli stessi acceleratori hardware. Il Politecnico di Milano sta attualmente collaborando con BSC per lo sviluppo di un framework di gestione delle risorse hardware che sarà utilizzato dal supercomputer per poter eseguire simultaneamente molteplici applicazioni concorrenti. Questo framework prende il nome di BarbequeRTRM. L'obiettivo di questo progetto è quello di integrare Barbeque con la libreria DLB (Dynamic Load Balancing), sviluppata da BSC, in particolare con il componente DROM (Dynamic Resource Ownership Management). DROM, infatti, è il componente software che permette di comandare lo scheduler del sistema operativo Linux del supercomputer, per assegnare dinamicamente specifici processori a specifici processi. Quest'ultimo componente è necessario per il corretto funzionamento del framework sul dispositivo di supercalcolo, poiché disporrà dei permessi di root sul sistema operativo della suddetta macchina, che invece non sono concessi a Barbeque.

2 Background

2.1 BarbequeRTRM

BarbequeRTRM è un gestore di risorse a run-time modulare ed estensibile, volto a gestire l'allocazione di risorse computazionali (CPU, GPU, memoria, acceleratori hardware) per molteplici applicazioni concorrenti. Esso offre la libreria bbque_rtlib per lo sviluppo di applicazioni integrate secondo l'*Adaptive Execution Model* (*AEM*), un modello di sviluppo software che guida l'applicazione attraverso un percorso di esecuzione ben definito, che viene orchestrato esternamente da Barbeque stesso. Tale percorso è caratterizzato da:

- *Resource-awareness*: l'applicazione è consapevole delle risorse assegnate e dunque in grado di riconfigurarsi in base ad esse
- *Runtime performance monitoring and negotiation*: l'applicazione è in grado di osservare il throughput corrente e chiedere l'assegnamento di ulteriori risorse

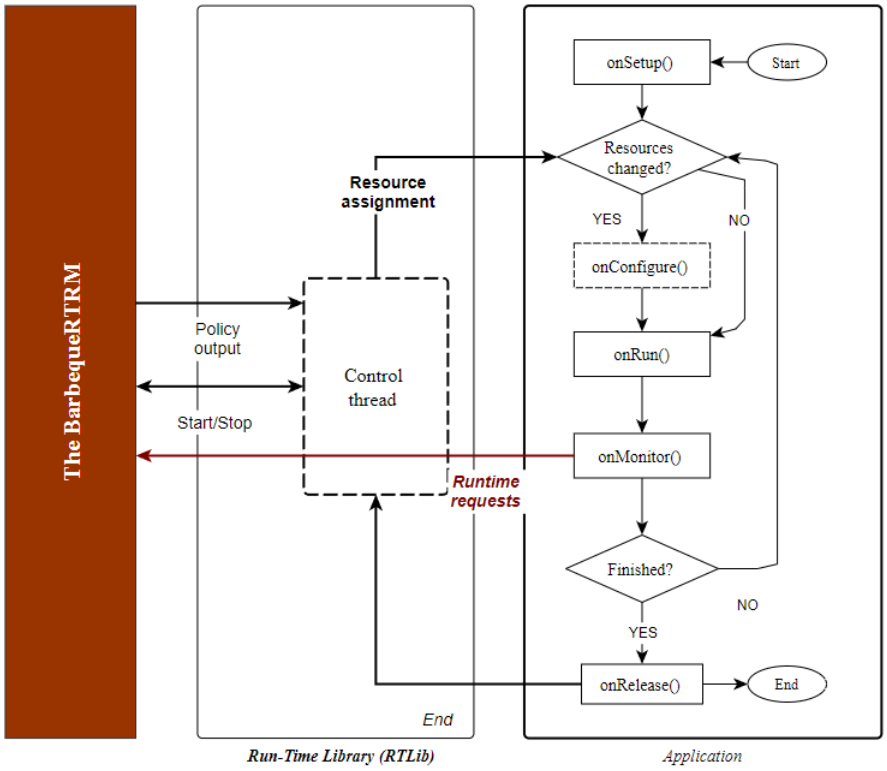

Figura 1: *Adaptive Execution Model*

Lo schema riportato in figura 1 mostra l'*Adaptive Execution Model*, che divide l'esecuzione dell'applicazione in diverse fasi:

- `onSetup()`: inizializzazione dell'applicazione (allocazione in memoria delle strutture dati necessarie all'esecuzione, inizializzazione delle variabili, etc...)
- `onConfigure()`: riconfigurazione in base alle risorse disponibili. Si verifica ogni qualvolta il thread di controllo di Barbeque assegna all'applicazione un nuovo set di risorse computazionali
- `onRun()`: esecuzione di un ciclo computazionale dell'applicazione
- `onMonitor()`: monitoraggio del throughput e richiesta di ulteriori risorse o riconfigurazione dell'applicazione volta ad ottimizzarne l'esecuzione
- `onRelease()`: chiusura dell'applicazione e de-allocazioni

2.2 DROM

DROM è un componente della libreria DLB che consente di riassegnare risorse computazionali tra diversi processi. DROM offre un API che può essere usata da entità esterne, come job scheduler e resource manager, per appropriarsi di una CPU utilizzata da un'applicazione in esecuzione e dedicarla a un'altra.

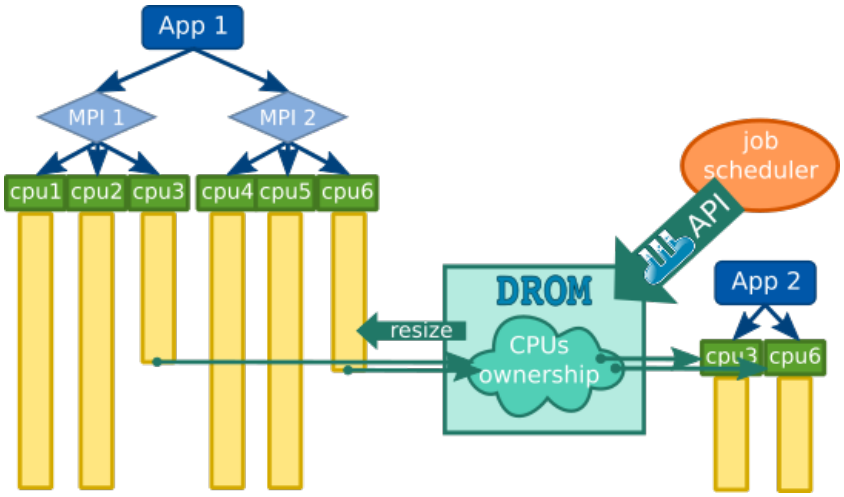

Figura 2: Funzionamento di DROM

Di seguito l'API messa a disposizione da DROM:

```
//Attach process to DLB as third party
int DLB_DROM_Attach(void)

//Detach process from DLB
int DLB_DROM_Detach(void)

//Get the total number of available CPUs in the node
int DLB_DROM_GetNumCpus(int *ncpus)

//Get the PIDs attached to this module
void DLB_DROM_GetPidList(int *pidlist, int *nelems,
    int max_len)

//Get the process mask of the given PID
int DLB_DROM_GetProcessMask(int pid, dlb_cpu_set_t
    mask, dlb_drom_flags_t flags)

//Set the process mask of the given PID
int DLB_DROM_SetProcessMask(int pid, const
    dlb_cpu_set_t mask, dlb_drom_flags_t flags)
```

Per il corretto funzionamento di DROM è necessario che un processo già esistente o creato appositamente per lo scopo si renda amministratore di DROM tramite la chiamata alla funzione `DLB_DROM_Attach()`. Il processo amministratore potrà dunque gestire l'assegnamento di CPU ad altri processi tramite la funzione `DLB_DROM_SetProcessMask()`, con l'unico requisito che questi ultimi siano stati inizializzati attraverso la chiamata di `DLB_Init()` (parte dell'API essenziale di DLB) e abbiano abilitato DROM tramite l'apposito `dlb_arg '--drom'`, da passare alla `DLB_Init()`. I processi in questione potranno poi ottenere le informazioni riguardo le nuove risorse assegnate grazie alla funzione `DLB_PollDROM()` e potranno terminare correttamente la loro esecuzione con la funzione `DLB_Finalize()` (entrambe parte dell'API essenziale di DLB).

3 Requisiti

Essendo DROM sviluppato internamente a BSC, esso sarà installato sulla nuova macchina di supercalcolo in modo tale da disporre dei permessi di root sul sistema operativo, potendo quindi funzionare correttamente. Al contrario, non essendo BarbequeRTRM un software sviluppato da BSC, esso non disporrà dei permessi di root, il che risulterà in una limitazione delle sue funzionalità. Barbeque, infatti, non potrà propriamente riassegnare le risorse ai diversi processi, ma potrà comunque mettere in atto le sue policy di distribuzione delle risorse andando poi a comunicare con DROM, che si occuperà dell'effettivo riassegnamento. Questo rende quindi necessario l'utilizzo dell'API che DROM fornisce per consentire la comunicazione tra le parti.

4 Architettura software e dettagli tecnici

La prima parte di questo progetto è stata dedicata a comprendere il funzionamento di Barbeque attraverso l'integrazione di un'applicazione di benchmark scritta in linguaggio C appartenente alla suite Rodinia. L'applicazione scelta è Back Propagation. Il porting di Back Propagation ha richiesto un previo lavoro di documentazione, non solo su BarbequeRTRM e sull'*AEM*, ma anche sull'applicazione stessa e sull'API OpenMP. La fase di documentazione sull'applicazione è stata necessaria per conoscerne il funzionamento e poterne suddividere il codice nelle fasi sopracitate. Invece, quella riguardo OpenMP è stata necessaria poiché tale libreria è utilizzata dall'applicazione per poter eseguire codice in parallelo. L'applicazione è stata dunque riscritta suddividendo la sua esecuzione in cicli di lavoro, per poter conoscere le CPU assegnate dinamicamente attraverso la `onConfigure()` e riconfigurare il proprio ciclo in modo da utilizzare un numero di thread pari al numero di CPU disponibili. L'architettura imposta dall'*AEM* ha comportato inoltre la necessità di salvare in memoria lo stato corrente dell'applicazione dopo ogni ciclo di lavoro. Un video che mostra alcuni aspetti importanti del codice e una demo del porting di Back Propagation

confrontata con l'applicazione originale è disponibile a questo link. Il codice del porting di Back Propagation è disponibile a questa repository git.

Una seconda parte del progetto è stata dedicata a comprendere il funzionamento di DROM, attraverso dei test autonomi costruiti a partire dalla documentazione fornita da BSC. Un esempio significativo tra i test eseguiti in autonomia è la scrittura di due brevi programmi C: il primo che inizializza DLB ed entra in un loop infinito in cui richiede a DROM un set di risorse aggiornato a intervalli regolari; il secondo che si fa amministratore di DROM e assegna un nuovo `cpu_set` al primo processo. Stampando su console il numero di processori disponibili per il primo programma a ogni iterazione si è potuto constatare che il primo programma risponde correttamente al riassegnamento delle risorse imposto dal secondo programma.
Di seguito è riportato il codice dei due programmi:

Programma 1:

```
int proc_nr = 0;
cpu_set_t cpu_set;
CPU_ZERO(&cpu_set);

if(DLB_Init(proc_nr, &cpu_set, "--drom") ==
   DLB_SUCCESS) printf("DLB correctly initialized\n");
else printf("DLB initialization FAILED\n");

printf("pid: %d", getpid());

while(1) {

    printf("Updating available processors: %s\n",
      DLB_Strerror(DLB_PollDROM(&proc_nr, &cpu_set)));
    printf("Number of available processors: &d",
      proc_nr);
    sleep(2);
}
```

Programma 2:

```
if(DLB_DROM_Attach() == DLB_SUCCESS) printf("Process
   attached to DLB as DROM administrator\n");
else printf("FAILED to attach process to DLB as DROM
   administrator\n");

//esempio: assegno al processo identificato da pid i
   primi 4 processori
int pid = ...;
int proc_nr = 4;
cpu_set_t cpu_set;
CPU_ZERO(&cpu_set);
for(int i = 0; i < proc_nr; i++) CPU_SET(i, &cpu_set);
DLB_DROM_SetProcessMask(pid, &cpu_set,
   DLB_STEAL_CPUS);
```

Infine, l'ultima parte del progetto consiste nella vera e propria integrazione tra BarbequeRTRM e DROM, sviluppata a partire dall'applicazione di benchmark precedentemente integrata con Barbeque. Per prima cosa è stato necessario individuare un processo da rendere amministratore di DROM. La scelta è ricaduta su Barbeque stesso, che in quanto resource manager sceglierà attraverso specifiche policy le CPU da assegnare ad ogni processo e le comunicherà successivamente a DROM tramite la `DLB_DROM_SetProcessMask()`. Per integrare l'applicazione è stato sufficiente inizializzare DLB nella `onSetup()` (senza richiedere alcuna risorsa iniziale, dunque passando 0 come parametro ncpus e un `cpu_set` vuoto come parametro mask) e chiamare `DLB_PollDROM()` nella `onConfigure()` per conoscere il nuovo set di risorse ogni qualvolta esso venga aggiornato da Barbeque. Infine è stato necessario finalizzare correttamente DLB nella `onRelease()` chiamando `DLB_Finalize()`. Tutte le modifiche apportate al porting di Back Propagation per consentire l'integrazione con DROM sono commentate e segnalate opportunamente all'interno del codice dell'applicazione. Le modifiche da apportare a Barbeque saranno invece gestite dal team di sviluppo dedicato, dopo una fase di verifica.

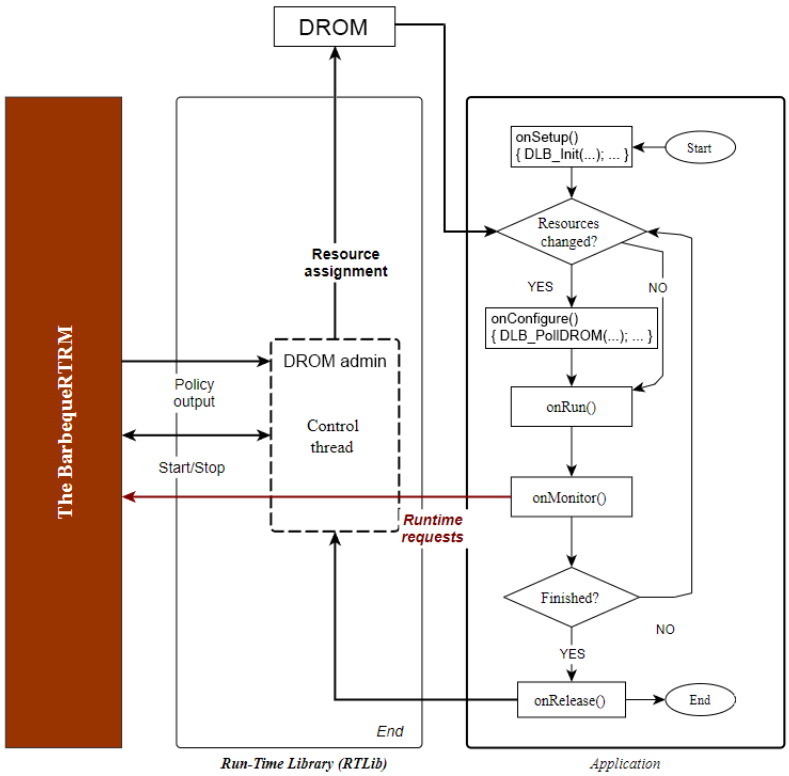

Figura 3: Architettura software dell'integrazione tra BarbequeRTRM e DROM

5 Risultati e sviluppi futuri

L'integrazione risultante è coerente con la documentazione di DROM fornita da BSC e con il funzionamento di Barbeque. Nonostante sia già stata parzialmente testata, essa richiede ulteriori verifiche sia da parte del team di sviluppo di BarbequeRTRM al Politecnico di Milano, che da parte degli sviluppatori di BSC. Una volta conclusa la fase di verifica su macchine standard, Barbeque dovrà essere adattato ulteriormente per eseguire sul supercalcolatore che verrà fisicamente realizzato nei prossimi quattro anni. In ultimo, saranno quindi necessarie delle verifiche finali sul funzionamento di BarbequeRTRM e della sua integrazione con DROM nel centro di supercalcolo di Barcellona.