Linux Checkpoint/Restart

Facciamoci un giro sulla macchina del tempo!

Micky Del Favero
micky@linux.it

BLUG - Belluno Linux User Group

LinuxDay 2011 - Belluno 22 ottobre 2011

Definizione di Checkpoint/Restart

Checkpoint salvataggio uno stato arbitrario di un processo

Definizione di Checkpoint/Restart

Checkpoint salvataggio uno stato arbitrario di un processo

Restart ripartenza di un processo da uno stato che un *checkpoint* ha precedentemente salvato

Come funziona?

Salva e ripristina uno stato di un processo in esecuzione

- file
- memoria
- IPC
- socket
- pty
- etc.

Come funziona?

Salva e ripristina uno stato di un processo in esecuzione

Opera a livello OS, granularità pari a task / container

Come funziona?

Salva e ripristina uno stato di un processo in esecuzione

Opera a livello OS, granularità pari a task / container

Simile a snapshot di VM, ma

- per processo in luogo di per sistema
- può essere più flessibile
- è più complicato
- non richiede supporto hardware e/o hardware virtualizzato

A cosa può servire?

Ripristino del processo ad uno stato precedente.

- Tolleranza ai guasti
  - Elaborazioni di lunga durata
  - HPC / Cloud
- Debug / Collaudo
  - ripartenza da stato stabile
  - ripetizione di passi specifici
  - distribuzione su più host
- Lancio rapido di applicazioni
- Macchina del tempo :-)

A cosa può servire?

Ripristino del processo ad uno stato precedente.
Sospensione di un processo e successiva ripartenza, anche su host diversi.

- Gestione della memoria
- Gestione del sistema
  - gestione tempo macchina
  - sospensione di applicazioni secondarie in caso di carico eccessivo
- Gestione sessione utente
  - sessione mobile
  - VPS / container

A cosa può servire?

Ripristino del processo ad uno stato precedente.
Sospensione di un processo e successiva ripartenza, anche su host diversi.
Migrazione di processi su host remoti.

- Bilanciamento del carico
  - HPC
  - Cloud
  - VPS
- Zero-downtime grazie a migrazione dei processi su server remoti
- Alta affidabilità
  - Backup live
  - Ripristino in caso di crash

Implementazioni

Distributed MultiThreaded CheckPointing (DMTCP)

- Checkpoint/Restart completamente in userspace
- Basato su *LD_PRELOAD*
- Non richiede nessuna patch!

Implementazioni

Distributed MultiThreaded CheckPointing (DMTCP)

Qemu / KVM

- Checkpoint/Restart in userspace
- Granularita pari ad un intero OS

Implementazioni

Distributed MultiThreaded CheckPointing (DMTCP)
Qemu / KVM
Berkeley Lab C/R (BLCR)

- Approccio ibrido kernel/userspace
- Richiede patch del kernel
- Progetto abbandonato?

Implementazioni

Distributed MultiThreaded CheckPointing (DMTCP)
Qemu / KVM
Berkeley Lab C/R (BLCR)
OpenVZ

- Checkpoint/Restart in kernel space
- Richiede patch del kernel

Implementazioni

Distributed MultiThreaded CheckPointing (DMTCP)
Qemu / KVM
Berkeley Lab C/R (BLCR)
OpenVZ
Linux-CR

- Checkpoint/Restart in kernel space con interfaccia di Restart in userspace
- Branch del kernel (*git clone...*)
- Usa cgroups

Implementazioni

Distributed MultiThreaded CheckPointing (DMTCP)

Qemu / KVM

Berkeley Lab C/R (BLCR)

OpenVZ

Linux-CR

DMTCP

DMCTP Distributed MultiThreaded CheckPointing

Completamente in userspace

Non richiede privilegi particolari

Non richiede moduli o modifiche al kernel né al software da controllare

Usato nell'esperimento CMS del Large Hadron Collider del CERN

Stato dell'arte

Gestisce automaticamente

- processi: fork, exec, ssh, pid, etc
- ipc: mutex, sem
- socket: unix, ipv4, ipv6
- terminali: pipe, pty, terminal mode, signal handler
- file: I/O, ownership, file condivisi, shm (via mmap)

Stato dell'arte

Supporta una vasta gamma di applicazioni, anche binarie, fra cui

- Matlab
- Perl, Python, shell
- VNC
- OpenMPI

Stato dell'arte

Sistemi non supportati

- Infiniband
- Myrinet

Funzionamento

Lancio del *coordinator*

Lancio del processo in *dmtcp*

Checkpoint (Salvataggio dell'immagine)

Termine del processo

Eventuale *Restart* da un *Checkpoint* precedente.

Esempio pratico

Lancio del coordinator

Esempio pratico

```
$ dmtcp_coordinator
```

Esempio pratico

```
$ dmtcp_coordinator
```

Lancio del processo in dmtcp

Esempio pratico

```
$ dmtcp_coordinator

$ dmtcp_checkpoint command -par0 ... -parN
```

Esempio pratico

```
$ dmtcp_coordinator
```

```
$ dmtcp_checkpoint command -par0 ... -parN
```

Checkpoint

Esempio pratico

```
$ dmtcp_coordinator
$ dmtcp_checkpoint command -par0 ... -parN
> c
```

Esempio pratico

```
$ dmtcp_coordinator
$ dmtcp_checkpoint command -par0 ... -parN
> c
Il processo termina
```

Esempio pratico

```
$ dmtcp_coordinator

$ dmtcp_checkpoint command -par0 ... -parN

> c
Il processo termina
Restart
```

Esempio pratico

```
$ dmtcp_coordinator

$ dmtcp_checkpoint command -par0 ... -parN

> c
Il processo termina

$ ls
ckpt_command_1fa17adc8b68249.dmtcp
dmtcp_restart_script_1fa17adc8b68249.sh
dmtcp_restart_script.sh
$ ./dmtcp_restart_script.sh
```

Linux-CR

Linux CR

Progetto iniziato da Oren Laadan e IBM

Punta ad essere integrato nel kernel ufficiale

Usato nei supercomputer PERCS / BlueWater

Obiettivi

Trasparenza da e verso i processi

Obiettivi

Trasparenza da e verso i processi

Affidabilità

- Checkpoint ok $\Rightarrow$ Restart
- Segnala situazioni non *Checkpoint*-abili
- Checkpoint non invasivo

Obiettivi

Trasparenza da e verso i processi
Affidabilità
Performance

- non ha impatto sulle performance del sistema
- il codice ha un'immagine regionevole

Obiettivi

Trasparenza da e verso i processi
Affidabilità
Performance
Mantenibilità

- codice generico in kernel/checkpoint
- codice dei sottosistemi in subsystem/checkpoint
- cambio versione non provoca grossi impatti (2.6.27 in avanti)

Stato dell'arte

Sottosistemi supportati

- tasks: thread, segnali, credenziali, etc.
- tutti i namespace tranne mount-ns
- sysvipc: shm, msg, sem
- file: regolari, fifo/pipe, epoll, eventi, dispositivi
- socket: unix, ipv4, ipv6
- sicurezza: smack, selinux

Stato dell'arte

Sottosistemi supportati

Sottosistemi non supportati

- lock, leases, owner
- file cancellati
- inotify/dnotify
- mount
- filesystem /proc

Stato dell'arte

Sottosistemi supportati
Sottosistemi non supportati
Architetture supportate

- i386
- x86-64
- s390x
- PowerPC
- ARM

Funzionamento: Checkpoint

1. Congela la gerarchia dei processi
2. Salva i dati
3. Salva la gerarchia dei processi
4. Salva lo stato di tutti i task
5. Scongela (o killa :-)) la gerarchia dei processi

Funzionamento: Restart

1. Crea il container
2. Crea la gerarchia dei processi
3. Ripristina lo stato di tutti i task
4. Riprendi l'esecuzione

Installazione

```
git clone git://www.linux-cr.org/pub/git/linux-cr
git clone git://www.linux-cr.org/pub/git/user-cr
git clone git://www.linux-cr.org/pub/git/tests-cr
cp /boot/config-‘uname -r‘ .config
$EDITOR. .config
make-kpkg --intrd kernel_image
cd user-cr && make all
cd test-cr && make all
```

Esempio pratico

```
mkdir /cgroup/mytask
echo $PID > /cgroup/mytask/tasks
echo FROZEN > /cgroup/mytask/freezer.state
checkpoint $PID > statefile
echo THAWED > /cgroup/mytask/freezer.state
restart --pidns < statefile
```

Documentazione, sorgenti, etc.

DMTCP

http://dmtcp.sourceforge.net/

Linux-CR

http://www.linux-cr.org/
http://www.linux-cr.org/pub/git/linux-cr
http://www.linux-cr.org/pub/git/user-cr
http://www.linux-cr.org/pub/git/tests-cr

Domande?

Grazie per l'attenzione!


Attribuzione Non commerciale Condividi allo stesso modo 3.0 Italia CC BY-NC-SA 3.0)