효율적인 원격 프로세스 검사점/재시작 프레임워크

최형준*, 김은성*, 정임영*, 염헌영*
*서울대학교 컴퓨터공학부
e-mail : {hjchoi, eskim, iyjung, yeom}@dcslab.snu.ac.kr

Efficient Remote Process Checkpoint/Restart Framework

Hyung Jun Choi*, Eunsung Kim*, Im Y. Jung*, Heon Young Yeom*
*Dept. of Computer Science, Seoul National University

요 약

병렬/분산 시스템의 신뢰도를 높이기 위해서는 결함 내성 기능을 갖추는 것이 필수적이다. 본 논문에서는 원격으로 프로세스에 대해 주기적으로 검사점을 만들고 예기치 않은 장애가 발생했을 경우 이를 신속히 가장 최근의 상태로 복원시켜 시스템의 안정성을 높일 수 있는 프레임워크에 대해서 연구한다.

1. 서론

병렬/분산 시스템의 신뢰도를 향상시키기 위해서는 결함 내성(fault-tolerance) 기능이 필수적이다. 시스템의 규모가 커질수록 시스템이 내재하는 결함요소가 늘어나게 되기 때문에 전체적인 시스템의 안정성이 현저히 줄어들게 된다[1]. 이러한 문제를 해결하기 위해 가장 많이 사용되는 기법 중 하나가 검사점(checkpointing) 기법이다[2]. 검사점 기법은 현재 실행되고 있는 프로세스의 상태를 주기적으로 파일에 저장해놓았다가 장애 발생시 프로세스의 가장 최근 상태로 임의의 시간에 복구할 수 있는 기법을 말한다.

본 논문에서는 병렬/분산 시스템 환경에서 수행되고 있는 프로세스를 원격으로 관리하고 예기치 않은 장애가 발생했을 경우 이를 신속히 감지하고 자동으로 가장 최근의 상태로 복구시킬 수 있는 원격 프로세스 검사점/복구 프레임워크에 대해서 연구한다.

2 장에서는 원격 프로세스 검사점/재시작 기법을 위한 전체적인 프레임워크 및 각각의 컴포넌트에 대해서 살펴보고 3 장에서는 실험환경 및 결과에 대해서 논의하고자 한다. 마지막으로 4 장에서는 결론을 맺도록 한다.

2. 프로세스 검사점/재시작 설계

병렬/분산 시스템 환경에서 효율적인 프로세스 검사점/재시작 기능을 제공하는 플랫폼을 설계하기 위해서는 다음과 같은 컴포넌트들이 필요하다:

Portable Batch System (PBS): PBS[3]는 프로세스를 시작하고, 감시하고, 종료시키고, 관리하기 위한 도구다. 기본적으로 모든 프로세스는 PBS 를 통해 시작된다.

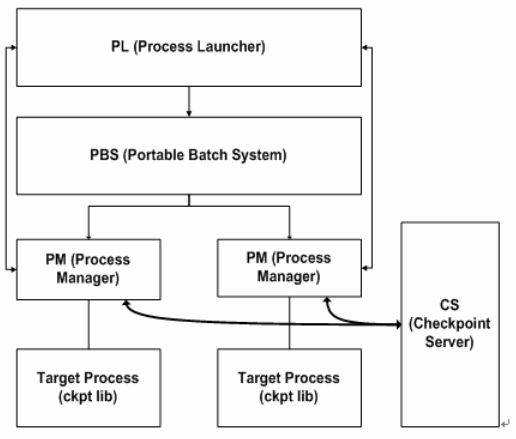
Process Launcher: 대상 프로세스를 실행시키기 위한 명세서를(예: 호스트, 환경 변수) PBS 에 전달하는 사용자 프로그램으로 대상 프로세스가 시작된 후 Process Manager 로부터 프로세스 관리 정보를 받고, Process Manager 에 대상 프로세스와 관련된 명령(검사점/재시작)을 내리는 역할을 담당한다.

Process Manager: PBS 로부터 대상 프로세스 명세서를 전달 받아 대상 프로세스를 기동하고, 관리한다. 관리 정보를 Process Launcher 에 전달하고, 대상 프로세스에 대한 명령을 Process Launcher 로부터 전달 받아 수행한다.

Checkpoint Library: Zandy[4]의 검사점/재시작 라이브러리를 수정한 것으로, Process Manager 와 IPC 통신을 통해 메시지를 주고 받으면서 프로세스의 검사점을 저장하는 부분이다.

Checkpoint Server: 대상 프로세스의 검사점 이미지 파일이 보관되는 서버다. 프로세스 장애 발생 시 Process Manager 의 요청에 따라 해당 프로세스의 검사점 이미지 파일을 Process Manager 에게 전송한다.

그림 1 은 앞서 설명한 컴포넌트들을 어떻게 구성하였는지를 보여주는 설계도이다.

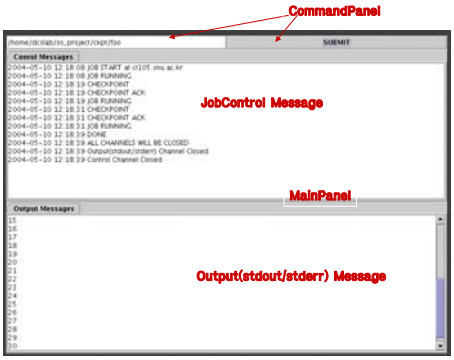


(그림 1) 원격 검사점/재시작 프레임워크

3. 구현

Process Launcher

Process Launcher 는 사용자가 직접적으로 다루어야 하는 부분이기에 사용자 편의성을 위해 그래픽 사용자 인터페이스(GUI)로 구현하였다. 또한 이식성을 고려하여 프로그래밍 언어는 Java 를 선택하였고 GUI 는 Java Swing 으로 구현하였다. 그림 2 는 Process Launcher 의 실행화면이다. 상단에는 프로세스를 실행시킬 수 있는 CommandPanel 이 있고 중간 부분에는 시스템의 상태 및 검사점 작업에 관한 기록을 출력하는 ControlMessagePanel 이다. 마지막으로 하단에는 원격에서 실행시킨 프로그램의 결과값을 출력하는 OutputPanel 이 있다. CommandPanel 을 통해 기술한 프로그램 명세서를 제출하면 PBS script 로 변환하여 PBS 서버를 호출하고 이에 따라 PBS 서버는 기술된 자료를 바탕으로 해당 Process Manager 를 fork/exec 하여 Process Launcher 와 TCP/IP 연결을 맺게 된다.



(그림 2) Process Launcher

Process Manager

Process Manager 는 대상 프로세스의 명세서를 전달받아 대상 프로세스를 시작하고, 관리한다. 또한 대상 프로세스의 실시간 관리 정보를 Process Launcher 에 전달하고 관련된 모든 명령을 Process Launcher 로부터 전달받아 수행하게 된다. Process Manager 는 C 로 작성하였으며 주요 함수는 다음과 같다:

Main: Process Manager 가 사용하는 모든 시그널(SIGCHLD, SIGUSR1, SIGUSR2)의 시그널 핸들러를 등록하고 Process Launcher 와 채널을 생성한다. 그리고 실행할 프로그램 명세를 Process Launcher 로부터 받아 수행하는 함수이다.

Sig_child_handler: 자식 프로세스가 종료될 때 발생하는 시그널을 처리한다. 즉, 정상적으로 종료하였는지 비정사적인 종료인지 확인하여 해당 작업을 수행하는 함수이다.

Sig_usr1_handler: 자식 프로세스는 fork/exec 된 후 Process Manager 에게 SIGUSR1 시그널을 보낸다. SIGUSR1 시그널을 받은 Process Manager 가 JOB RUNNING 메시지를 Process Launcher 에게 전송하는 함수이다.

Sig_usr2_handler: 자식 프로세스가 검사점을 수행한 후 Process Manager 에게 SIGUSR2 시그널을 보낸다. SIGUSR2 시그널을 받은 Process Manager 가 CHECKPOINT 메시지를 Process Launcher 에게 전송하는 함수이다.

Fork_and_exec: 실행 명세를 받아 프로세스를 실행하는 함수이다.

Checkpoint Client/Server

검사점 이미지 파일을 저장, 관리해주는 서버 프로세스는 Java 로 구현하였으며 기본적으로 클라이언트의 요청에 따라 TCP/IP 연결을 맺어 사용자 요청을 처리하는 스레드를 시작하도록 구현하였다.

4. 실험

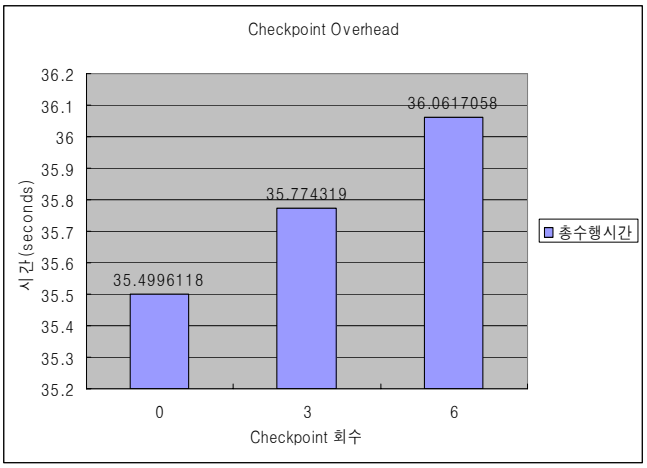
실험환경은 P4 1.8GHz, 512MB 메모리 기계 세 대를 사용하였다. 하나는 Process Launcher 로 사용하였고 나머지 두 대는 PBS, 검사점 서버(checkpoint server), 그리고 Process Manager 로 사용하였다. 대상 프로세스로는 LU factorization 을 수행하는 프로그램을 사용하였다.

실험의 흐름은 다음과 같다:

1) Process Launcher 가 LU factorization 을 위한 명세서를 PBS script 로 변환하여 PBS 서버를 호출한다.
2) PBS 는 Process Manager 를 구동하고, Process Manager 는 LU factorization 프로그램을 fork/exec 한다.
3) Process Manager 는 Process Launcher 와 TCP/IP 연결을 맺고 실패여부를 보고한다.
4) Process Launcher 는 사용자가 설정한 사용주기에 맞춰 Process Manager 에 검사점 명령을 TCP 로 전달한다.
5) 검사점 명령을 전달 받은 Process Manager 는 IPC(SIGUSR1)를 사용하여 LU factorization 프로세스에게 전달한다.
6) Process Manager 는 검사점 이미지 파일을 검사점 서버에 전송하고 성공여부를 Process Launcher 에 전달한다.

이 시스템은 프로세스 자체의 잘못된 연산과 프로세스가 수행되는 기계의 고장으로 인해 발생하는 장애를 감지하여 재시작 메커니즘을 수행한다. 프로세스의 잘못된 연산으로 인한 장애가 발생하면 Process Launcher 에 프로세스 실패 보고서를 보낸다. Process Launcher 는 프로세스 재시작을 위한 명세서를 다시 PBS script 로 변환하여 PBS 서버에 요청한다. PBS 는 Process Manager 를 시작하고 Process Manager 는 검사점 서버에 가장 최근 검사점 이미지 파일을 요청한다. Process Manager 는 이를 이용하여 이전 상태로 복구하여 프로세스를 재시작 한다. 하지만 물리적으로 기계가 고장이 나서 발생한 장애의 경우에는 Process Manager 가 감지할 수 없으므로 Process Launcher 가 직

접 감지할 수 있게 설계한다. Process Launcher 는 Process Manager 와의 TCP/IP 연결이 끊기게 되므로 Process Manager 로부터 프로세스 실패 통지가 없으면 "기계고장"으로 간주해 다른 기계로 이미지를 이주시켜 실행하게 된다.



(그림 3) 체크포인트 회수

그림 3 은 검사점 회수에 따른 총 수행시간의 변화를 보여준다. 실험을 위해 LU factorization 의 검사점 주기를 다르게 하여 각각 0,3,6 회 수행하게 했으며, 각각 총 다섯 번 실험하여 그 평균값을 구했다. 검사점을 잡지 않고 LU factorization 을 수행했을 때의 총 수행시간은 35.49 초이며, 3 회 검사점을 잡으면 35.77 초이고, 6 회 잡으면 36.06 초로 검사점 시간이 약 0.1 초도 안됨을 알 수 있다. 이 때, 검사점 이미지 파일의 크기는 1338KB 였다.

5. 결론

프로세스 검사점/재시작 기법은 프로세스의 검사점 이미지 파일을 사용하여 프로세스가 수행 중에 장애가 일어나도 적절한 검사점으로부터 다시 시작하게 해서 낭비하게 되는 연산자원을 최소화하게 해주는 방법이다.

본 논문에서는 원격 호스트에서 프로세스를 시작하고, 장애가 감지됐을 때 검사점/재시작 기법을 사용하여 원격에서도 프로세스를 다시 시작하게 하는 시스템 프레임워크를 구현하였다.

참고문헌

[1] Taesoon Park, Heon Y. Yeom, "Application Controlled Checkpointing Coordination for Fault Tolerant Distributed Computing Systems", Parallel Computing
[2] E. N. Elnozahy and D. B. Johnson and Y. M. Wang, "A survey of rollback-recovery protocols in message-passing systems", ACM Computing Surveys
[3] http://www.openpbs.org
[4] http://pages.cs.wisc.edu/~zandy/ckpt/