제목: 튜플 스페이스를 갖는 란다모델의 설계 및 사양 기법

이건영, 최민호, 원영선, 윤만표
아주대학교 정보 및 컴퓨터공학부

Design and Mapping of Hierarchical Tuple Space Linda Model

Geonyoung Lee, Minho Choi, Youngsun Weon, Manpyo Hong
Aju University Information & Computer Engineering

8. 약

변구조적 병렬 휴먼모델의 란다 모델은 미결합성, 확장성, 분산작동, 동기화 문제 해결과 같은 많은 특성을 가지고 있는 반면, 튜플 스페이스의 메시지 집중 형식으로 설계 시작의 장점을 하락시키는 결정적인 문제를 앵다. 따라서 본 논문에서는 개체적 튜플 스페이스를 제정함으로써 기존 란다 모델의 장점을 모두 수용하고 동시에 핵심적 결정을 줄일 수 있는 계층적 튜플 스페이스 란다 모델을 제안한다. 또한 계층적 튜플 스페이스는 구현이 가능한 병렬 코드 생성 방법을 제공하고, 그 과정에서의 문제점 및 프로그램 성능 분석은 실제 실시간에 결합함으로써 진전적인 프로세스와 효과적 스크립트를 가능하게 하는 사양 기법을 제안한다.

1. 서론

고등 컴퓨터의 개발과 병렬 프로그램 개발의 보편화에 따라 비용 상여 메시지의 성능을 높이 버치하고 있 다. 그 이유 중 하나는 병렬 작업의 복잡성 외부에 인한 개인 트래픽이 덜어지는 경향을 보이고 있기 때문이다. 따라서 다른 병렬 계산 방법은 병렬 컴퓨터 구조와 몽리교적(architecture-independent) 방 속구조의 개신 보일(UPC/MOSAIC Universal Parallel Computational Model)로서의 방향을 허용하여 기존의 순차 컴퓨터에서의 오노난 개인 메시지 번역에 달성된 문제를 해결하기 위하여 제안되었다. הה행 기준의 병렬 계산 모델은 병렬 컴퓨터 구조로 몽리교적이고(architecture-independent) 방 속구조의 개신 보일(UPC/MOSAIC Universal Parallel Computational Model)로서의 방향을 허용하여 기존의 순차 컴퓨터에서의 오노난 개인 메시지 번역에 달성된 문제를 해결하기 위하여 제안되었다. 허행 기준의 병렬 계산 모델로 수행하기에 적합하지 않기 때문에 이 단 허행 기준의 병렬 모델로 수행하기에 적합하지 않기 때문에 이 단

2. HTS 란다 모델

기존의 란다 모델은 한계적 HTS 란다 모델은 튜플, 튜플 스페이스, 튜플 매칭, 기본 연산과 같은 기존 란다 모델의 모든 수행하고 본속적인, 튜플과 튜플의 개념, 개체적 튜플 스페이스, 구조의 인원이 추가된 형태이다.

2.1 모듈(Module)과 태스크(Task)

HTS 란다 모델에서는 하나의 복합적인 란다 프로그램을 모듈로, 하나의 모듈을 구성하는 프로세스들을 튜플로 정의한다. 즉 모듈은 하나의 튜플 스페이스를 통해 여러 프로세스들이 서로 비디오를 주고 받으며 수행하며 나가는 프로그램을 의미한다. 하나의 모듈은 하나의 튜플 스페이스와 여러 튜플들을 가지며 문제를 해결하니 새로운 클라우드 모듈은 다시 서로 프로세스로 나누어질 수 있지만, 이때 각 서브 모듈들 역시 하나의 튜플 모델이 된다.

2.2 계층적 튜플 스페이스

단다 모델에서 프로그램을 생성하기 위해서는 반드시 개체의 시간이 언어의 하나이며, 이에 따라 튜플 스페이스의 데이터 요구는 몽리교적 튜플 스페이스(Hierarchical Tuple Space - HTS) 란다 모델을 제안한다. 이를 제안한 HTS 란다 모델은 기존 란다 모델의 튜플 스페이스를 적합성을 향상시키기 위해 새로운 튜플 스페이스로 나누고 각각의 튜플 스페이스를 병렬로 수행하도록 명령어 테이크주기, 트래픽을 절감하고 하여 공간적 효율을 높일 수 있다. 또한 HTS 란다 모델을 위한 병렬 프로그래밍 방법을 도입하고 실제 테스트와 동작하며 설명적으로 분산적인 프로세스의 효율적 수행을 가능하게 하는 방법을 제공한다.

론문은 1992년 한국학회에서 발표한 논문(N95-100-19-01-33)의 연구에 일부를 차용하였다.

![그림 1. 계층적 튜플 스페이스](image)
으로 이루어져 있고, 모든 프로그램은 조합법의 초기 품목 스테이크를 남겨 놓고 각 다분 프로그램에서 사용한다.

2.3 부가적 연산
HTS 런던 모델은 그 특성상 학습 모델로서의 실제 결과를 부가적으로 스테이크로 남기는 작업을 필요로 하므로 부가적 연산의 추가를 요한다. 따라서, 학습의 빌드 모델에서 두 모듈 스테이크로 결과 품목들을 넘겨받는 경우에 사용되는 연산 $out_{up}$을 추가하였다.

3. HTS 런던 모델을 위한 벨포드 생성
HTS 런던 모델에서는 하나의 프로세스에서 여러 개의 모듈로 구성되며, 각각의 모듈들은 하나의 벨포드 프로그램을 의미하도록 구성되어 있다. 또한 런던 모델에서는 실제 품목 생성에 사용되는 부품들을 업데이트하기 위해서 미리 소스 프로그램을 처리한 후에 품목 생성을 하는데 data dependence analysis의 결과를 바탕으로 부품들의 생성이 바탕화된 프로세스 (parallelism)를 찾아내 자동적 스테이크 이론에 품목 생성 과정에서 사용되는 모듈을 생성하는 방법으로 구성하였다. 실제로 품목 생성 과정에서 이러한 끝단의 인터페이스의 형성 (Dependence 관계)를 찾기 위해 벨포드를 의미하게 소스 프로그램을 생성하여, 이에 새로운 생성된 소스 프로그램을 생성하여 비교, 이에 따라 생성된 코드와 결과가 OCCAM에서 사용하는 품목 생성이 이루어지며, 이에 생성된 소스 프로그램은 협업형 HTS 런던 모델에 대한 생성을 하며, 이를 통해 만들어진 파트이자 품목 스테이크를 만들어 준다.

3.1론리적 포함의 함의 프로그램에 대한 벨포드 생성
평행 프로그램 생성은 각 단계로 이루어진다. 각 단계에서는 전체 소스 프로그램을 네트워크마다 (macro instruction) 단위로 구동하여 매개 변수와 해시된 데이터 중심으로 구성된 결과를 생성한다. 여기에서 데이터는 모드와 기준으로 구축한 결과를 포함한다. 각 단계에서는 이러한 데이터를 미리 데이터 중심으로 구축하여 메모리로 저장한 프로그램 구성을 하는데 이차하여 데이터 중심으로 구성한다.

평행 코드 생성 과정은 여러 단계로 이루어진다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 여기서 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중심 데이터 중심으로 구성된 결과를 생성한다. 각 단계에 대한 데이터 중
지 모델에서의 기기 시간이 결정되면 그 결과에 의해 프로세서 핵심
트레이스트 생성하고, 실제 프로세서 시각에 사용 한다.
4.1 생성된 각 모델의 실행시간 예측
각 모델의 실행시간을 예측하기 위해 심층 분석기는 프로파일러
(profiler), 경직 상황 분석(state performance predictor), 동적 상황 예
(dynamic performance predictor) 방식으로 진행된다.
프로파일러는 프로그램을 실행시동으로 각 문장의 실행시간을 구
하여 프로파일 자료(pdata)를 만들인다. 이는 각 문장의 실행
시간을 구하기 위해서는 반드시 프로그램 전체에 맑다면 것은 아
니다. 즉 프로그램의 축적함성(speed up)을 위해 각 프로그램을 실
현하기 위해 최소한의 문장만이 필요하다. 따라서 최소한의 문
장만을 구하기 위해 프로그램 실행이버 버그를 사용하며, 이를
이 때 HTS 라인 트레이스를 입력으로 받음에서 생성된
문장 분석 결과를 활용한다.
신경 계획하는 프로파일 자료를 첨조로 각 변선의 실행시간과 평균시간
을 예측하여 기계 설계해더(machined description table)를 만든고 그 데
이를 이용하여 설계 상의 자료를 제공한다.
품목 설계하는 실제 기계설계 실행시기 시장에 패와하는 방법으
로 프로파일 자료와 경직 상황 자료를 입력으로 받는 설계 상의
자료를 생성하게 된다. 따라서 동적 분석기는 프로파일러의 크기가 큰
경우에는 실행할 수 있으며 너무 많은 시간이 소요되며, 스케린
(skeleton) 프로그램을 생성함으로써 이 문제를 해결하였다. 즉, 스케
목은 프로그램 실행시기의 전체 프로그램을 실행 시켜볼 수 없을 때
있당 가능한 성능과 거의 동일한 성능을 보장함과 동시에 시간 20
~30배로 매우 속도 정량 결과를 얻을 수 있다.
스케린 프로그램은 다음에 가장 다름과 같다.
(1) 프로파일자료를 받아 '필 여부'를 변환한다. 프로파일자료
수는 프로파일자료의 횟자로 생성한 프로파일자료를 토대로 한다. 동시에
문장이 있는 경우 문장수를 실행 횟수만 만 정확하게 한다. 조건문
의 실행횟수 역시 프로파일자료의 횟자로 한다. 즉, 프로파일자료의
의회가 조건문을 만족할 경우 프로파일자료의 조건문 안 문장수를 횟수
에 환산함으로 나타난다. 그런데 원래 회의가 조건문이 없고
문장 프로그램은 현재의 프로그램의 동일한 실행시간을 요구하
려고 하면 흔히 동일한 실행시간이 요구하다고
그리고 이에 따라 프로파일 프로그램 생성 방법에 대해 연구가 진행 중에 있다.
4.2 사과
상속 분석기의 이상 각 모델의 실행시간이 정확히 몇 걸그룹에 의해
해 각 모델당 산출된 수의 해답이 생성한다.
(1) HTS는 랜덤 모델의 성능을 통해 각 모델의 실행시간에 예측하여 각 모델의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
(2) HTS는 랜덤 모델의 조건문의 실행시간에 의해 고정시점의 프로그램의
이기 때문에 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 트레스로는 실행시간을 일정하게 하며 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.
따라서 각 랜덤 모델의 조건문의 실행시간에
에 의해 고정시점의 프로그램의 수의 해답을 산출한다.