@NUMA hinting page faults

각 태스크의 메모리는 주기적으로 Runtime에 Unmap 된다.

태스크가 Unmap된 메모리에 접근하려면 Page fault 발생 이것이 NUMA hinting page faults

@NUMA page migration

NUMA page fault 는 Remote memory 부터 Migration Cost 보다 적다.

Migration의 기준을 동일한 Page fault가 두번 발생할때 하도록 정의

Migration 노드는 해당 프로세스가 동작하고 있는 노드

@Fault statistics

태스크를 이동시키기 위해 사용(CPU follows memory에 기반한 방법.)

&Statistics 는 각 태스크 마다 가지고 있음

&각 Node마다 NUMA page fault 횟수를 카운팅

&page migration될 경우 새로운 new location에 대한 새로운 카운트

@Type of NUMA faults

NUMA fault의 타입은 locality와 private vs shared로 나뉠 수 있다.

&Local vs Remote

Local fault : 태스크와 같은 노드에 있는 메모리에 대한 Page fault

Remote fault : 태스크와 다르 노드에 있는 메모리에 대한 Page fault

&Private vs Shared

Private fault : 동일한 태스크에 의해 연달아 2번 접근된 메모리

Shared fault : 마지막 접근한 태스크와 다른 태스크에 의해 발생한 fault

@Task placement

태스크를 어느 노드에 위치시켜야 할지 문제는 간단하지 않다.

메모리를 공유하는 경우가 많으며 노드에서 메모리에 접근하는 태스크들은 노드의 CPU코어에서 수행할 수 있는 수보다 많은 스레드를 가지고 있을 수 있다.

따라서 로드 밸런서는 빠른 처리를 위해 스레드들을 다른 노드로 분산시킨다.

Task placement는 로드 불균형을 만들어 내서는 안된다. 접근해야할 데이터가 이 노드에 있다고 집중된 로드를 분산 시키지 못하게 해선 안된다.

N개의 노드, 노드 K에 태스크 a가 수행중, Task placement는 태스크 a에 대해 모든 노드 체크. 어느 노드에 태스크 a의 가장 큰 메모리 부분을 가지고 있는지 점검

만약 노드 K보다 다른 노드 A에 더 많은 메모리 부분이 존재한다면 노드 A가 idle인지 확인, idle이 아니라면 A에서 수행중인 태스크 t가 K에서 더 잘 수행 될 수 있는지 확인

또한 태스크 t를 노드 K로 옮기는게 이득이 되는지 확인

@Task grouping

공유 메모리 사용시 멀티 태스크들은 동일 메모리에 접근할 수 있다. 공유메모리를 찾기위해 PID와 CPU 번호를 사용

NUMA task placement를 향상시키기 위해 NUMA fault가 발생하였을때 페이지가 마지막으로 Fault난 CPU를 확인하고 PID가 동일하다면

Task grouping group으로 만듭니다. group은 연관된 태스크로 구성

이 group을 이용해 Task placement와 함께 NUMA balancing을 향상

@Task grouping & task replacement

group stat은 그룹내 task의 stat의 합

task stat보다 group stat을 사용하면 shared memory에 대해 보다 효율적인 액세스가 가능하다

만약 태스크가 numa group에 속하고 다른 task가 이 numa\_group으로 속하도록 한다면 효율적 태스크 배치 가능

@Pseudo-interleaving

가끔 하나의 워크로드가 여러 노드에 분산된 경우가 존재

이런경우 Memory bandwidth를 최대화 하고 태스크가 사용하는 Private 메모리를 유지하면서 페이지 마이그레이션을 최소화 해야함

Private fault를 항상허용, Shared Fault는 더 로드가 큰 노드에서 적은 노드로만 페이지 마이그레이션을 허용하게 됨(즉 한쪽 메모리 로드만 높은 상황을 피함)