Program. paralela/distribuida

Entornos de programación paralela basados en modelos/paradigmas

Índice

- Reflexiones sobre la programación paralela
- MapReduce
  - Propuesta original de Google
  - Introducción a la programación Mapreduce en Hadoop
  - Diseño de algoritmos con MapReduce
- Otros modelos de programación paralela
  - BSP
  - Basados en grafos (Pregel)

Introducción entornos progr. paralela

- Valoración OpenMP y MPI
  - Programación de bajo nivel
  - Baja productividad
  - Programador debe centrarse en numerosos aspectos complejos (planificación, sincronización, comunicación…) ajenos al problema
  - Destinados a programadores expertos
- ¿Herramientas de programación más productivas, para programadores de propósito general?
  - Entornos de programación paralela basados en modelos (paradigmas, patrones) que resuelvan, y oculten al programador, los aspectos de bajo nivel y le permitan centrarse en el problema
    - P.ej. en universo Google: MapReduce o Pregel

Paralelismo para las masas

- Actualmente sistemas paralelos por doquier
  - Multi-*cores,* GPUs, Many-*cores,* Plataformas *cloud* paralelas
- Empujado por tecnología no por demanda desarrolladores SW
  - Límites constructivos en mejoras del rendimiento de un procesador
    - *ILP wall; Power wall; Memory wall*
  - ¿Qué hacer con el creciente nº transistores que nos da ley de Moore?
    - Múltiples procesadores en un chip
  - Además de ILP (*Instruction-Level Parallelism*)
    - Proporcionan TLP (*Thread-Level Parallelism*)
  - ILP → paralelismo implícito → transparente a la aplicación
  - TLP → paralelismo explícito
- ¿Cómo desarrollar programas que aprovechen paralelismo?
  - ¿Herramientas de extracción automática? ¿Nuevos lenguajes?

Buenas noticias: 30 años de paralelismo

Sistemas paralelos tienen ya una larga tradición

- Supercomputadores vectoriales (SIMD)
- SMP/UMA (MIMD con memoria compartida)
- NUMA (MIMD con memoria compartida)
- MPP (MIMD con memoria distribuida)
- Clusters (MIMD con memoria distribuida)
- Grids (MIMD con memoria distribuida)

Malas noticias: 30 años de software

Experiencia después de 30 años

- Extracción automática de paralelismo:
  - Aplicación muy limitada
- "Lenguajes" específicos para paralelismo:
  - Enorme fragmentación
    - OpenMP (m. compartida).y MPI (m. distribuida) más usados
  - Alta complejidad
  - Poca productividad
  - Uso restringido a expertos
    - No adecuados (ni pensados) para programadores de propósito general
- Pero paralelismo se ha extendido a todos los ámbitos:
  - Se requiere herramientas adecuadas para todo tipo programadores
  - Solución: ¿Nuevos lenguajes? → No: Mejor nuevo enfoque

Entornos programación paralela en los 90

ABCPL
ACE
ACT++
Active messages
Adl
Adsmith
ADDAP
AFAPI
ALWAN
AM
AMDC
AppLeS
Amoeba
ARTS
Athapascan-0b
Aurora
Automap
bb_threads
Blaze
BSP
BlockComm
C*.
"C* in C
C**
CarlOS
Cashmere
C4
CC++
Chu
Charlotte
Charm
Charm++
Cid
Cilk
CM-Fortran
Converse
Code
COOL

CORRELATE
CPS
CRL
CSP
Cthreads
CUMULVS
DAGGER
DAPPLE
Data Parallel C
DC++
DCE++
DDD
DICE.
DIPC
DOLIB
DOME
DOSMOS.
DRL
DSM-Threads
Ease .
ECO
Eiffel
Eilean
Emerald
EPL
Excalibur
Express
Falcon
Filaments
FM
FLASH
The FORCE
Fork
Fortran-M
FX
GA
GAMMA
Glenda

GLU
GUARD
HAsL.
Haskell
HPC++
JAVAR.
HORUS
HPC
IMPACT
ISIS.
JAVAR
JADE
Java RMI
javaPG
JavaSpace
JIDL
Joyce
Khoros
Karma
KOAN/Fortran-S
LAM
Lilac
Linda
JADA
WWWinda
ISETL-Linda
ParLin
Eilean
P4-Linda
Glenda
POSYBL
Objective-Linda
LiPS
Locust
Lparx
Lucid
Maisie
Manifold

Mentat
Legion
Meta Chaos
Midway
Millipede
CparPar
Mirage
MpC
MOSIX
Modula-P
Modula-2*
Multipol
MPI
MPC++
Munin
Nano-Threads
NESL
NetClasses++
Nexus
Nimrod
NOW
Objective Linda
Occam
Omega
OpenMP
Orca
OOF90
P++
P3L
p4-Linda
Pablo
PADE
PADRE
Panda
Papers
AFAPI.
Para++
Paradigm

Parafrase2
Paralation
Parallel-C++
Parallaxis
ParC
ParLib++
ParLin
Parmacs
Parti
pC
pC++
PCN
PCP:
PH
PEACE
PCU
PET
PETSc
PENNY
Phosphorus
POET.
Polaris
POOMA
POOL-T
PRESTO
P-RIO
Prospero
Proteus
QPC++
PVM
PSI
PSDM
Quake
Quark
Quick Threads
Sage++
SCANDAL
SAM

pC++
SCHEDULE
SciTL
POET
SDDA.
SHMEM
SIMPLE
Sina
SISAL.
distributed smalltalk
SMI.
SONiC
Split-C.
SR
Sthreads
Strand.
SUIF.
Synergy
Telegrphos
SuperPascal
TCGMSG.
Threads.h++.
TreadMarks
TRAPPER
uC++
UNITY
UC
V
ViC*
Visifold V-NUS
VPE
Win32 threads
WinPar
WWWinda
XENOOPS
XPC
Zounds
ZPL

Mattson y Keutzer (UCB EES): *Patterns for Parallel Programming*

Lista de deseos

Herramientas de desarrollo de *apps*. paralelas deberían:

- Ser productivas
- Facilitar la programación de aplicaciones paralelas
- Posibilitar el desarrollo de aplicaciones
  - Correctas
  - Eficientes
  - Escalables
  - Portables
- Para programadores de propósito general:
  - Más importante productividad y facilidad de desarrollo que eficiencia
  - Para expertos puede ser más prioritaria la eficiencia

Dificultades de la programación paralela

- Para empezar las de la programación concurrente
  - *Why Threads Are A Bad Idea (for most purposes)* de J. Ousterhout
- El programador debe enfrentarse (entre otros) a:
  - Encontrar la concurrencia de la aplicación: identificación de tareas
    - Minimizar partes no paralelizables
    - Buscar granularidad adecuada: +fina → + paralelismo pero + sobrecarga
  - Asignación de tareas a procesadores y planificación de las mismas
    - Dificultad para un reparto de carga adecuado
      - Una aplicación no se completa hasta final de la última tarea
  - Establecer esquemas de sincronización y comunicación entre tareas
    - Controlar sobrecarga evitando interbloqueos y condiciones de carrera
  - Tolerancia a fallos
    - Ejecución involucra múltiples nodos: alta probabilidad de que uno falle
    - ¿Qué hemos estudiado al respecto en OpenMP y MPI?
  - E/S: Aplicaciones *Big Data* requieren E/S masiva
    - ¿Cómo distribuir entradas y salidas de la aplicación?

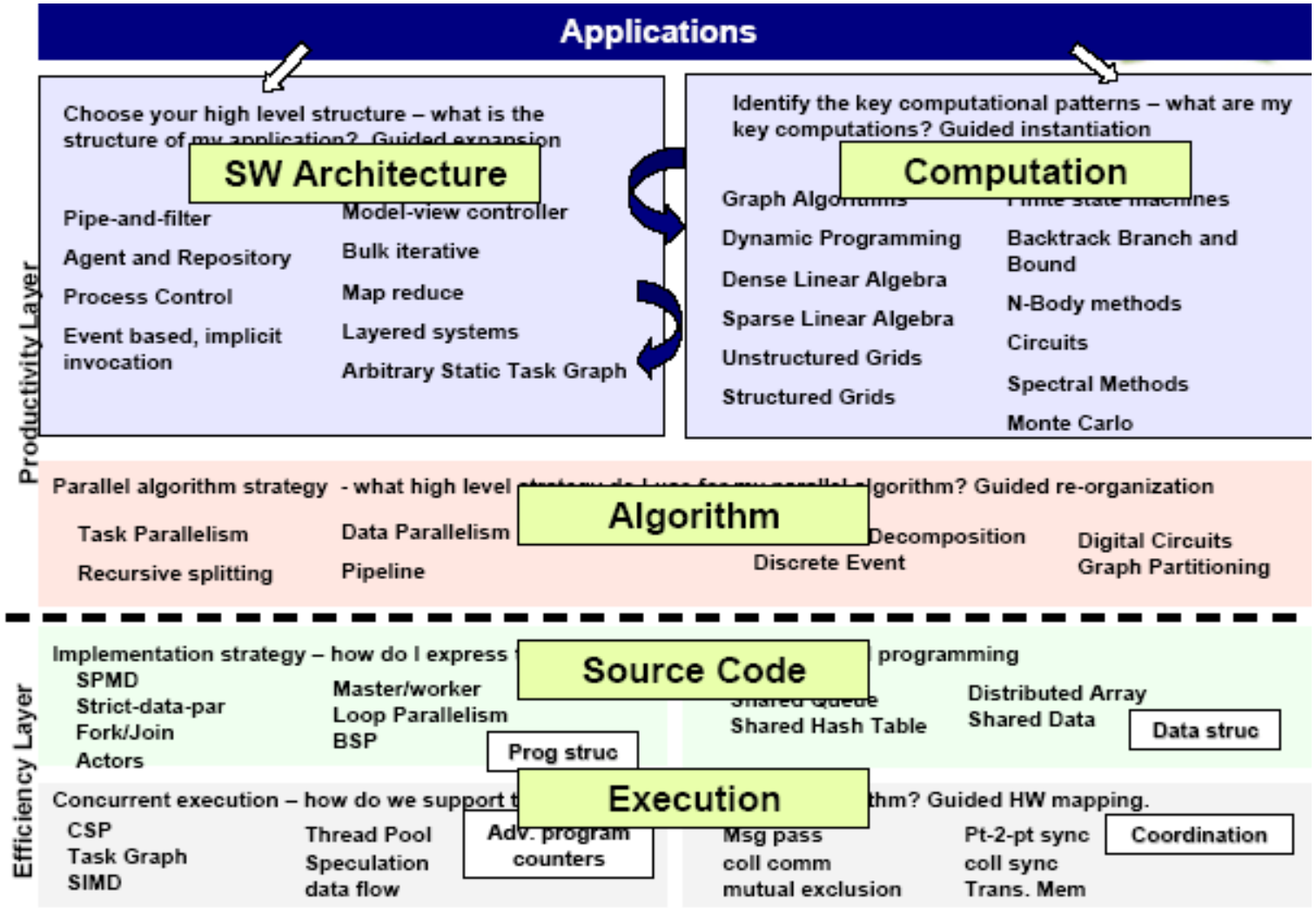
Modelos de programación paralela

- Modelo abstracto de la máquina paralela
  - Facilita el desarrollo de programas paralelos
  - Independiente de la arquitectura subyacente
    - p.e. MPI sobre sistema de m. compartida
    - p.e. OpenMP sobre cluster con DSM
- Distintas alternativas en el diseño de un modelo tales como:
  - Memoria compartida (OpenMP) vs. paso de mensajes (MPI)
    - M. compartida → + fácil de programar pero + difícil de poner a punto
  - SPMD (*Single Program Multiple Data*) vs. MPMD
    - SPMD basado en SIMD
    - SPMD basado en multithread (OpenMP)
    - SPMD una copia del programa en cada nodo (MPI)
  - Paralelismo datos (*omp for*) vs. paralelismo de tareas (*omp sections*)

Patrones de programación paralela

- Patrones exitosos en diseño de software
- Investigadores en Intel y Berkeley proponen usar *Parallel Programming Patterns:* Proyecto OPL
- Plantea 5 categorías jerárquicas de patrones (4 niveles):
  - Patrones estructurales (nivel más alto; más abstracto):
    - Describen la organización global de la aplicación
  - Patrones computacionales (nivel más alto; más abstracto):
    - Describen los tipos de computaciones que realiza la aplicación
  - Patrones de estrategia de algoritmos paralelos (2º nivel):
    - Describen estrategias de alto nivel para explotar la concurrencia
  - Patrones de estrategia de implementación (3º nivel):
    - Elementos incluidos en código del programa para expresar paralelismo
  - Patrones de ejecución paralela (nivel más bajo):
    - Mecanismos que dan soporte a la ejecución de aplicaciones paralelas

Estructura de OPL versión 2



Frameworks de programación paralela

- Modelo/patrón especifica pauta de construcción/programación
- Pero además puede posibilitar desarrollo de *framework* con funcionalidad que dé soporte al modelo facilitando programación
  - Aunque - flexible y – eficiente si problema no encaja en modelo
  - P.e. modelo/patrón *fork-join*:
    - Programador con *pthreads* puede usarlo pero tiene que programarlo
    - OpenMP lo implementa: programación más sencilla
- Propuesta:
  - Capa eficiente: programadores expertos implementan soporte de un modelo ocultando aspectos bajo nivel (planificación, sincronización, comunicación, tolerancia a fallos...)
  - Capa productiva: programadores de propósito general desarrollan como en un sistema no paralelo centrándose en el problema
- MapReduce y Pregel encajan en esa idea: modelo y *framework*