granularidad

Se suele distinguir concurrencia

- de grano fino
  es decir, se aprovecha de la ejecución de operaciones concurrentes a nivel del procesador (hardware)
- a grano grueso
  es decir, se aprovecha de la ejecución de procesos o aplicaciones a nivel del sistema operativo o a nivel de la red de ordenadores

clases de architecturas

Una clasificación clásica de ordenadores paralelos es:

- SIMD (*single instruction multiple data*)
- MISD (*multiple instruction single data*)
- MIMD (*multiple instruction multiple data*)

procesadores modernos

- Hoy día, concurrencia a grano fino es estándar en los microprocesadores.
- El la familia de los procesadores de Intel, por ejemplo, existen las instrucciones MMX, SSE, SSE2, SSE3, SSSE3 etc. que realizen según la clasificación SIMD operaciones en varios registros en paralelo.
- Ya están en el marcado los procesadores con multiples núcleos, es decir, se puede programar con varios procesadores que a su vez puedan ejecutar varios hilos independientes.

graphics processing units (GPU)

La programación paralela y concurrente (y con pipeline) se revive actualmente en la programación de las GPUs (graphics processing units) que son procesadores especializados para su uso en tarjetas gráficas que cada vez se usa más para otros fines de cálculo numérico.
Los procesadores suelen usar solamente precisión simple.

conmutación

multi–programación o *multi–programming*: los procesos se ejecutan en hardware distinto

multi–procesamiento o *multi–processing*: Se aprovecha de la posibilidad de multiplexar varios procesos en un solo procesador.

multi–tarea o *multi–tasking*: El sistema operativo (muchas veces con la ayuda de hardware específico) realiza la ejecución de varios procesos de forma cuasi–paralelo distribuyendo el tiempo disponible a las secuencias diferentes (*time–sharing system*) de diferentes usuarios (con los debidas medidas de seguridad).

computación en red

- La visión de ‘computación en la red’ no es nada más que un gran sistema MIMD.
- Existe una nueva tendencia de ofrecer en vez de aplicaciones para instalar en el cliente, una interfaz hacia un servicio (posiblemente incorporando una red privada virtual) que se ejecute en una conjunto de servidores.
- Existe una nueva tendencia de usar un llamado GRID de superordenadores para resolver problemas grandes (y distribuir el uso de los superordenadores entre más usuarios).

mecanismos de conmutación

Existen dos puntos de vista relacionados con el mecanismo de conmutación

- el mecanismo de conmutación es *independiente* del programa concurrente
  (eso suele ser el caso en sistemas operativos),
- el mecanismo de conmutación es *dependiente* del programa concurrente
  (eso suele ser el caso en sistemas en tiempo real),

En el segundo caso es imprecindible incluir el mecanismo de conmutación en el análisis del programa.

mecanismos de conmutación

- Al desarrollar un programa concurrente, no se debe asumir ningún comportamiento específico del planificador (siendo la unidad que realiza la conmutación de los procesos).
- No obstante, un planificador puede analizar los programas concurrentes durante el tiempo de ejecución para adaptar el mecanismo de conmutación hacia un mejor rendimiento/equilibrio entre usuarios/procesos (ejemplo: automatic “nice” en un sistema Unix).
- También los sistemas suelen ofrecer unos parámetros de control para influir en las prioridades de los procesos que se usa como un dato más para la conmutación.

memoria compartida homogéneo (SMP)

- Sin una memoria compartida no existe concurrencia (se necesita por lo menos unos registros con acceso común).
- Existen varios tipos de arquitecturas de ordenadores (académicos) que fueron diseñadas especialmente para la ejecución de programas concurrentes o paralelos con una memoria compartida (por ejemplo los proyectos NYU, SB-PRAM, o Tera)
- Muchas ideas de estos proyectos se encuentran hoy día en los microprocesadores modernos, sobre todo en los protocolos que controlan la coherencia de los cachés.

memoria compartida homogéneo

- La reordenación automática de instrucciones,
- la división de instrucciones en ensemblador en varios fases de ejecución,
- la intercalación de fases de instrucciones en los procesadores
- el procesamiento en pipelines,
- la aparencia de interrupciones y excepciones,
- la jerarquia de cachés

son propiedades desafiantes para la programación concurrente correcta con procesadores modernos y la traducción de conceptos de alto nivel del lenguaje de programación a código ejecutable no es nada fácil (y no igual en todos los entornos).

memoria compartida heterogéneo

- Sin embargo, no hace falta que se ejecute un programa en unidades similares para obtener concurrencia.
- La concurrencia está presente también en sistemas heterógenos, por ejemplo, aquellos que solapan el trabajo de entrada y salida con el resto de las tareas (discos duros).

comunicación y sincronización

La comunicación y sincronización entre procesos funciona

- mediante una memoria compartida (*shared memory*) a la cual pueden acceder todos los procesadores a la vez o
- mediante el intercambio de mensajes usando una red conectando los diferentes procesadores u ordenadores, es decir, procesamiento distribuido (*distributed processing*).

Sin embargo, siempre hace falta algún tipo de memoria compartida para realizar la comunicación entre procesos, solamente que en algunos casos dicha memoria no es accesible en forma directa por el programador.

sistemas híbridos

También existen mezclas de todo tipo de estos conceptos, por ejemplo, un sistema que use multi–procesamiento con hilos y procesos en cada procesador de un sistema distribuido simulando una memoria compartida al nivel de la aplicación.

algunas herramientas para C/C++ (no exhaustivas)

- ACE (Adaptive Communications Environment)
  `http://www.cs.wustl.edu/~schmidt/ACE.html`
  (usa patrones de diseño de alto nivel, p.ej.; proactor)
- Intel Concurrent Collections (C++, 2012, versión 0.7)
  `http://software.intel.com/en-us/articles/intel-concurrent-collections-for-cc/`
  (lenguaje de preprocesado para expresar concurrencia)
- Cilk/Cilk++/Cilk Plus (2011)
  `http://software.intel.com/en-us/articles/intel-cilk-plus/`
  (extensión de C/C++ para expresar concurrencia)
- Intel Thread Building Blocks (2012, version 4.0)
  `http://threadingbuildingblocks.org/`
  (C++ template librería para expresar concurrencia)

algunas herramientas para C/C++ (no exhaustivas)

- OpenMP
  `http://openmp.org/wp/`
  (C-preprocessor (+librería embebido) para expresar concurrencia)
- Noble (`www.non-blocking.com`)
- Qt threading library (`http://doc.qt.nokia.com/`)
- pthreads, boost::threads, Zthreads (uso directo de programación multi-hilo)
- próximo/ya estándar de C++ (C++11, 2011)
  (`http://www.open-std.org/jtc1/sc22/wg21/`)

Usa la Red para buscar más información, aquí un ejemplo.

Trazado de rayos concurrente

Rayon con OpenMP

modelo de memoria

- para programar a alto nivel (p.ej. Java. C++) se necesita un modelo de memoria consistente para tratar casos como el siguiente:

```
initially: a=b=0;

P0:                          P1:
   a: r1 = a;                   r3 = b;
   b: r2 = a;                   a = r3;
   c: if( r1==r2 )
   d:    b=2
```

modelo de memoria

- Una escritura de una variable `volatile` en Java garantiza que todas las escrituras a variables que aparecen antes en el código se hayan ejecutado ya.
- Entonces otro hilo ve un estado bien definido del hilo escribiendo una variable volatile.
- Es decir, se puede hablar de una relación *ha-pasado-antes* respecto a las escrituras y lecturas.
- Java implementa este modelo, pero es bastante restrictivo respecto a posibles optimizaciones automáticas del compilador.
- C++11 implementa un modelo, que es más flexible para el programador, una primera versión está disponible en g++.4.7 `http://gcc.gnu.org/wiki/Atomic/GCCMM`

Memoria Transaccional

```
P0:
atomic transaction { c = a - b; }

P1:
atomic transaction { if (a > b) b++; }
```

Está garantizado que `c` nunca es negativa.

Memoria Transaccional con C++

- la última versión del compilador C++ de GNU (g++-4.7, 2012) da soporte experimental de memoria transaccional.
- `http://gcc.gnu.org/wiki/TransactionalMemory`

comentarios

- Un ingeniero informático no solo usa herramientas ya existentes, sino adapta aquellas que hay a las necesidades concretas y/o inventa nuevas herramientas para avanzar. Eso se llama innovación tecnológica.
- En el ámbito de la concurrencia son más importantes los conceptos que las tecnologías dado que las últimas están actualmente en un proceso de cambio permanente.
- Se debe comparar un programa concurrente/paralelo con el mejor programa secuencial (conocido) para evaluar el rendimiento.