




# Taller de Programación Paralela

Fernando R. Rannou  
Departamento de Ingeniería Informática  
Universidad de Santiago de Chile

April 24, 2015



Escalabilidad

**Escalabilidad**



# Pintando una casa

Escalabilidad



- Supongamos que deseamos pintar una casa que tiene 10 habitaciones. Si me demoro 1 día por habitación, la casa completa me demorará 10 días.
- Si contrato un equipo de 10 personas, cada una podría pintar una pieza y la casa estaría terminada en poco más de 1 día; no exactamente 1 día, pues el equipo demorará un pequeño tiempo en asignación de piezas, distribución de tarros de pintura, etc.
- Si contrato 20 personas, la casa estará terminada en un poco más de 1/2 día.
- Si contrato 200?
  - ◆ Mayor tiempo empleado en asignación
  - ◆ Mayor tiempo de coordinación de las personas en cada pieza.

# Límite al speedup

Escalabilidad

- *Para un problema particular el Speedup tiene un límite*
- Dado un problema, por ejemplo multiplicación de matrices de  $100 \times 100$ , a medida que el número de procesadores aumenta, 5, 10, 100, se obtienen cada vez mejores speedups, pero llegará el momento en que la adición de nuevos procesadores no mejorará el speedup.
- Algunos factores que limitan el speedup son:
  - ◆ Overhead de software
  - ◆ Des-balance de carga
  - ◆ Overhead de comunicación/sincronización
  - ◆ Overhead secuencial que no puede paralelizarse
- La única forma de mejorar el speedup es aumentar el tamaño del problema a medida que se adicionan más procesadores

# Ley de Amdahl

Escalabilidad

- Supongamos que deseamos resolver un problema que tiene una porción *serial*, es decir no puede paralelizarse, y otra paralela.
- Si la porción secuencial demora  $T_s$  en un procesador y la porción paralela demora  $T_p$ , también en un procesador, entonces, el speedup alcanzable por  $n$  procesadores es:

$$S(n) = \frac{T_s + T_p}{T_s + T_p/n} \quad (1)$$

- Note que la definición de speedup de Amdahl corresponde a la noción de speedup relativo.

# Ejemplo en Amdahl

Escalabilidad

- Sumar dos matrices de  $N \times N$  demora:
  - ◆  $3N^2$ : dos lecturas y una escritura a disco
  - ◆  $N^2/(Vn)$ , sumar  $N^2$  números con  $n$  procesadores de velocidad  $V$
- El speed relativo es entonces:

$$\frac{3N^2 + N^2/V}{3N^2 + N^2/(Vn)} = \frac{3 + 1/V}{3 + 1/(Vn)}$$

- Cuando  $n \rightarrow \infty$ , el speedup es  $1 + 1/3V$

# Ley de Amdahl (2)

Escalabilidad

- Sea  $f = T_s / (T_s + T_p)$ , es decir la fracción secuencial del total del programa
- El speedup puede ser reescrito como

$$\begin{aligned} S(n) &= \frac{T_s + T_p}{T_s + T_p/n} \\ &= \frac{1}{\frac{T_s}{T_s + T_p} + \frac{T_p}{(T_s + T_p)n}} \\ &= \frac{1}{f + \frac{(1-f)}{n}} \\ &\leq \frac{1}{f} \end{aligned}$$

- Es decir, el speedup alcanzable está limitado por su parte secuencial

# Otra mirada a la Ley de Amdahl

Escalabilidad

- Amdahl concluyó que los computadores paralelos no podrían entregar speedups cada vez mayores, pues el speedup estaba limitado por la porción secuencial de los problemas, independientemente del computador paralelo!
- Por ejemplo, si el 5% de un programa es serial, el speedup máximo alcanzable es 20.
- Esto es verdad, cuando la fracción secuencial  $f$  no puede modificarse.
- Si  $0 < c \leq f$ , entonces la ley de Amdahl es aplicable, pero existen muchos casos donde la *carga de trabajo* puede modificarse, de tal forma que la parte secuencial se puede hacer muy, muy pequeña respecto de la paralela.
- En dichos casos, el speedup puede no tener límite.
- *La ley de Amdahl es una ley de speedup para cargas fijas de trabajo.*





# Speedup para cargas de trabajo variable

Escalabilidad



- En ciertas aplicaciones es importante obtener soluciones con una alta resolución y/o precisión, sin importar que el problema se demore mucho.
- Por ejemplo, la calidad de la solución de una ecuación diferencial por elementos finitos depende del número de nodos del dominio; si el aumento de número de nodos no aumenta sustancialmente la parte secuencial, el problema es candidato a tener speedups cada vez más grandes
- Otro ejemplo, son los algoritmos iterativos, donde la carga de trabajo depende del error de la solución y por lo tanto del número de iteraciones del algoritmo
- En estos y otros casos la ley de Amdahl debe ser modificada para considerar el aumento de la carga de trabajo

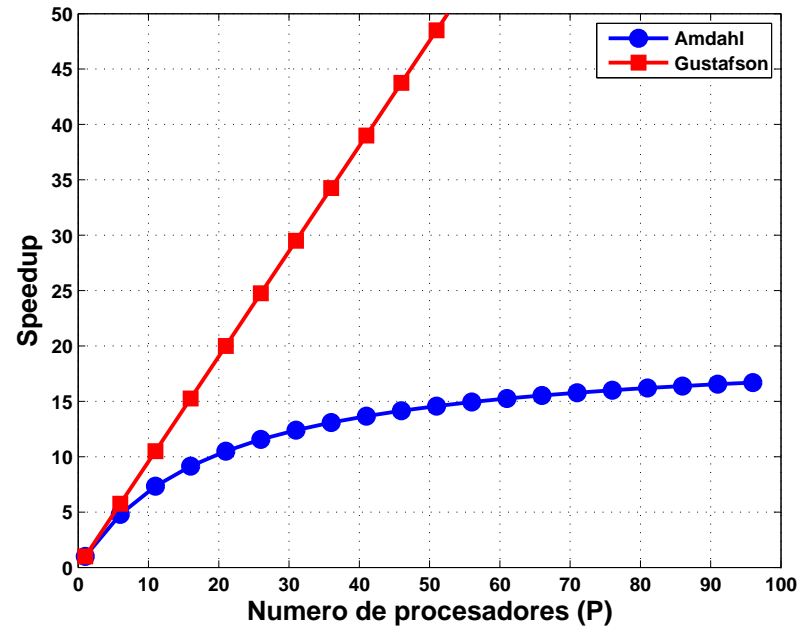
# La Ley de Gustafson

Escalabilidad

- Cuando la fracción paralelizable es escalable con  $n$ , obtenemos un speedup *escalable*

$$S(n) = \frac{T_s + nT_p}{T_s + T_p}$$

- Note que el tiempo secuencial crece con el número de procesadores, pero el tiempo paralelo permanece constante.





# Escalabilidad

Escalabilidad



- Recordamos que un *sistema paralelo* consiste en un problema y una arquitectura paralela:
- Un sistema paralelo es *escalable* cuando su rendimiento continúa mejorando en la medida que el tamaño del *sistema* crece, es decir
  - ◆ crece el tamaño del problema que queremos resolver (o su precisión)
  - ◆ y crece el número de procesadores del computador paralelo
- Como medida de rendimiento podemos usar, por ejemplo, speedup relativo para carga variable.

# Eficiencia (1)

Escalabilidad

- La *Eficiencia* de un sistema paralelo se define como:

$$E(n) = \frac{S(n)}{n}$$

- La eficiencia es una medida del grado de speedup alcanzado en relación al máximo.
- Para el caso de cargas fija,

$$E(n) \leq 1$$

es decir, cuando la porción serial es nula, el speedup es máximo,  $n$ , y la eficiencia es 1

- Pero para cargas de trabajo escalables con el número de procesadores, es posible

$$E(n) > 1$$

# Eficiencia (2)

Escalabilidad

- Para proseguir hagamos las siguientes definiciones:

1.  $T_1$ : tiempo total monoprocesador (secuencial)
2.  $n$ : número de procesadores
3.  $T_0$ : overhead total del programa paralelo (no existe en el secuencial)
4.  $T_n$ : tiempo total paralelo

- Luego,  $T_n = (T_1 + T_0)/n$ , y el speedup es

$$S(n) = \frac{T_1}{T_n} = \frac{nT_1}{T_1 + T_0}$$

- Luego, la eficiencia es  $E(n) = S(n)/n \Rightarrow$

$$\begin{aligned} E(n) &= \frac{T_1}{T_1 + T_0} \\ &= \frac{1}{1 + \frac{T_0}{T_1}} \end{aligned}$$

- En ciertos sistemas paralelos  $T_0 < 0$ , logrando un *speedup superlineal*

# Isoeficiencia

Escalabilidad

- Cuando el número de procesadores aumenta y el tamaño del problema (o la carga de trabajo) es el mismo, la tasa de aumento del speedup disminuye y por lo tanto la eficiencia también disminuye
- Pero con un número fijo de procesadores, a medida que la carga aumenta, el speedup también aumenta

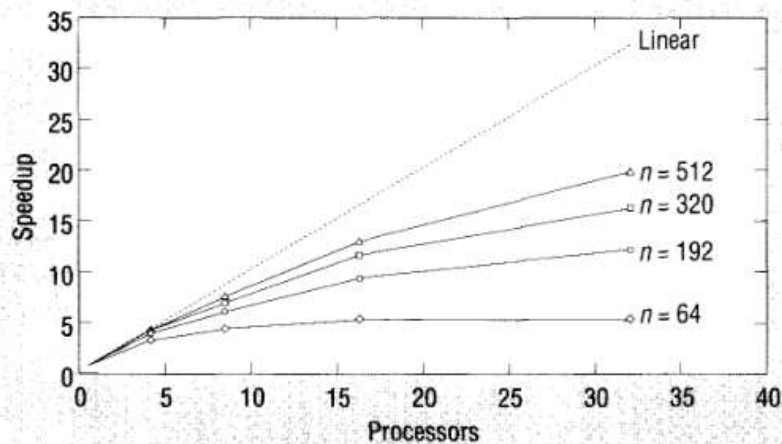


Table 1. Efficiency as a function of  $n$  and  $p$  for adding  $n$  numbers on  $p$ -processor hypercubes.

	$p = 1$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
$n = 64$	1.0	.80	.57	.33	.17
$n = 192$	1.0	.92	.80	.60	.38
$n = 320$	1.0	.95	.87	.71	.50
$n = 512$	1.0	.97	.91	.80	.62

Isoefficiency: measuring the scalability of parallel algorithms and architectures, A.Y. Grama, A. Gupta, and V. Kumar, IEEE Parallel and Distributed Technology, 1993

# Isoeficiencia (cont)

Escalabilidad

- ¿Con qué tasa debiera aumentar el tamaño del problema con respecto al número de procesadores para mantener la eficiencia constante?
- La *Isoeficiencia* es un índice de escalabilidad de un sistema, el cual mide la capacidad de un programa paralelo para utilizar efectivamente un número creciente de procesadores en una arquitectura dada.
- La función de isoeficiencia es una función de un algoritmo paralelo (programa computacional) y arquitectura paralela. Luego, un programa paralelo puede mostrar diferentes isoeficiencias (escalabilidad) en diferentes arquitecturas paralelas.
- La isoeficiencia relaciona el tamaño del problema al número de procesadores necesarios para mantener la eficiencia constante o para aumentar el speedup proporcionalmente al número de procesadores.

# Isoeficiencia (cont)

Escalabilidad

Sea  $T_0(W, n)$  el overhead incurrido por un programa paralelo cuando utiliza  $n$  procesadores para resolver un problema con carga de trabajo  $W$

El overhead paralelo (generalmente) aumenta cuando aumenta el número de procesadores para resolver un problema con carga fija  $W$ . Es decir, cuando  $W$  es constante,  $T_0(W, n)$  crece a medida que  $n$  aumenta.

■ Luego,

$$E(n) = \frac{1}{1 + \frac{T_0(W, n)}{T_1}}$$

donde  $T_1$  es el tiempo secuencial (sin overhead), el cual a su vez es proporcional a  $W$ .

■ Entonces, si  $t_c$  es el costo de ejecución de una operación.

$$E(n) = \frac{1}{1 + \frac{T_0(W, n)}{W t_c}}$$

■ A medida que  $n$  crece, y  $W$  es fijo,  $T_0(W, n)$  crece y por lo tanto la eficiencia disminuye



# Isoeficiencia (cont)

Escalabilidad

- Pero para algunos sistemas paralelos, si  $W$  aumenta y  $n$  es fijo, la eficiencia aumenta, pues el overhead  $T_0$  aumenta con una tasa menor a  $O(W)$ .
- Para estos sistemas se puede mantener la eficiencia constante a medida que se usan más procesadores y se aumenta la carga de trabajo
- **Estos sistemas son denominados escalables**
- Si

$$E = \frac{1}{1 + \frac{T_0(W,n)}{Wt_c}}$$

entonces, para que  $E$  sea constante,  $T_0/(Wt_c)$  debe ser constante. Luego,

$$\begin{aligned}\frac{T_0}{W} &= t_c \left( \frac{1-E}{E} \right) \\ W &= \frac{1}{t_c} \left( \frac{E}{1-E} \right) T_0\end{aligned}$$

# Isoeficiencia (cont)

Escalabilidad

- Si  $K = E/(t_c(1 - E))$ , entonces

$$W = KT_0(W, p)$$

- A través de manipulación algebraica se puede obtener  $W$  en función de  $n$ , con parámetro  $E$ , es decir  $f_E(n)$  es la función de isoeficiencia del sistema paralelo para mantener una eficiencia de  $E$ .
- **Sistemas altamente escalables:** Una función pequeña de isoeficiencia (constante  $K$  pequeña) significa que pequeños incrementos de carga de trabajo son suficientes para usar un número creciente de procesadores, eficientemente.
- **Sistemas pobremente escalables:** es necesario incrementar mucho la carga de trabajo respecto de un pequeño incremento en el número de procesadores, para mantener una eficiencia constante

# Ejemplo teórico de isoeficiencia

Escalabilidad

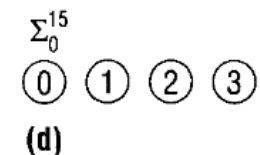
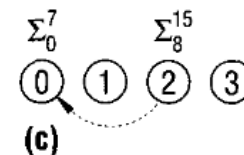
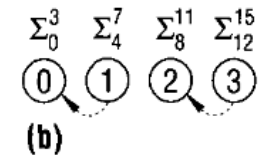
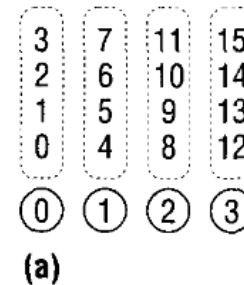
## Sumar $n$ números

En un computador secuencial, la carga de trabajo es justamente el número de operaciones y es igual a  $n$  (en realidad  $n - 1$  sumas). Luego

$$T_1 = nt_c$$

En un hipercubo de  $p$  procesadores

- Cada procesador suma  $n/p$  números, resultando  $p$  sumas parciales
- La suma de estos números se realiza enviando la suma parcial al nodo más cercano y sumando a la suma parcial local.
- Asumiendo que la comunicación de un número requiere 1 unidad de tiempo, entonces la suma de los  $p$  resultados parciales toma  $\log p$  unidades de comunicación y  $\log p$  unidades de suma



# Ejemplo teórico de isoeficiencia

Escalabilidad

Entonces, el tiempo paralelo es  $T_p = (n/p + \log p + \log p)$

$$T_p = \frac{n + p \log p + p \log p}{p}$$
$$T_p = \frac{T_1 + T_0}{p}$$

Es decir, el overhead paralelo es  $T_0 = 2 \log p$

**Speedup:**

$$S = \frac{T_1}{T_p} = \frac{n}{\frac{n}{p} + 2 \log p}$$

**Eficiencia:**

$$E = \frac{S}{p} = \frac{n}{n + 2 \log p}$$

# Ejemplo teórico de isoeficiencia

Escalabilidad

Finalmente,

$$W = K2 \log p$$

es decir la función de isoeficiencia es  $O(\log p)$

Si el número de procesadores aumenta de  $p$  a  $p'$ , la carga de trabajo (en este caso  $n$ ) debe aumentar en un factor de  $(\log p')/(\log p)$  para mantener constante la eficiencia