Calcolo parallelo e distribuito mod.B

Introduzione al corso
Prof. Marco Lapegna

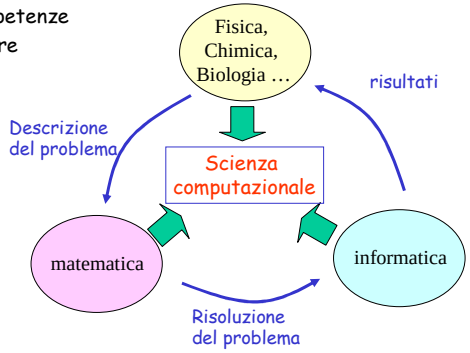
- Lun 8.30 – 10.30 aula E3
- Gio 8.30 – 10.30 aula E4
- prerequisiti:
  - superato il corso di Calcolo Numerico
  - aver scritto almeno un programma parallelo

Metodi dell'indagine scientifica

- Metodi tradizionali
  - Teoria (risoluzione carta e penna di equazioni)
  - Sperimentazione (costruzione di prototipi)
- Limitazioni:
  - Troppo difficile (es: costruire grandi gallerie del vento)
  - Troppo costoso (es: costruzione di prototipi usa e getta)
  - Troppo lento (es: studio di evoluzione di galassie o clima)
  - Troppo pericoloso (es: armi, virus, esperimenti climatici)
- Terza metodologia: Simulazione (uso di calcolatori)

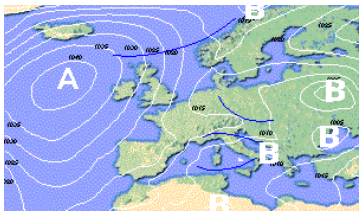
Computational science

- Studio dei fenomeni scientifici attraverso la simulazione computazionale
- 3 distinte competenze
- Interdisciplinare



Esempio: previsioni meteorologiche

Superficie = 100 milioni di Km^2
Altezza s.l.m. = di 10 Km
Discretizzazione = cubi di lato 100m ($1Km^3 = 10^3$ cubi)



$100 \times 10^6\ Km^2 \times 10\ Km = 10^9\ Km^3$
$= 10^9 \times 10^3 = 10^{12}$ cubi

Esempio (cont.)

- previsioni per i prossimi 2 giorni (48 ore)
- ogni ora di simulazione in ogni cubo necessarie 10^4 operazioni floating point (flop)

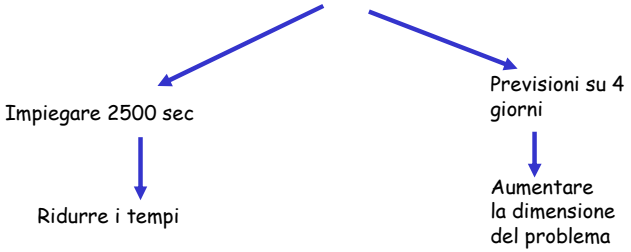
Intera superficie = $10^{12} \times 10^4 \times 48 =$
$\sim 5 \times 10^{17}$ operazioni

Calcolatore da 100 Tflops (100×10^{12} op/sec)

~ 5000 sec (circa 1.5 ore)

Esempio (cont.)

- Con un calcolatore da 200 Tflops e' possibile



Obiettivi della computational science:
Risolvere problemi piu' complessi
in minor tempo

Campi di applicazione

- Scienza
  - Modelli climatici
  - Modelli astrofisici
  - Biologia: proteine, nuovi farmaci
  - Nuovi materiali
- Ingegneria
  - Crash test
  - Terremoti e modelli strutturali
  - Disegno di aeroplani
  - Disegno di motori
- Economia
  - Modelli finanziari
  - Motori di ricerca
  - Data mining
- Altro
  - Simulazione di test nucleari
  - Crittografia

Per tali problemi sono richieste potenze di calcolo di almeno 1 Pflops (10^{15} op/sec)

Utilizzo dei supercalcolatori oggi

- Un supercalcolatore e' un unico sistema (hardware e software) capace di fornire una potenza di calcolo paragonabile a quella massima che e' possibile ottenere con la corrente tecnologia
- E' una definizione che dipende dal tempo (Un supercomputer di oggi e' un computer ordinario di domani)
- Nell'accezione comune un calcolatore e' un supercomputer se rientra nella Top500 list.

La Top500 list

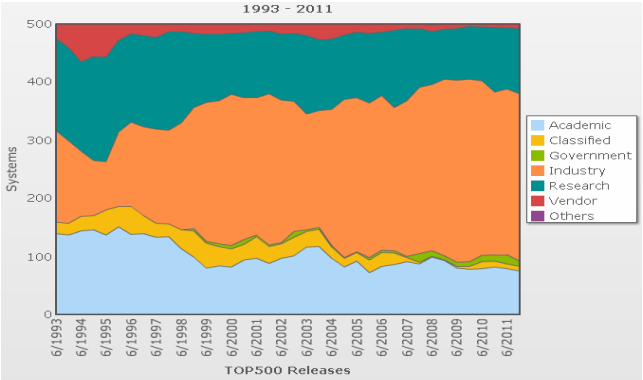
- Elenca i 500 supercomputer piu' potenti al mondo
- Lista aggiornata 2 volte all'anno
- Performance misurata con risoluzione di sistemi di equazioni lineari (Linpack benchmark)
- www.top500.org

La Top500 list: novembre 2011

Top five

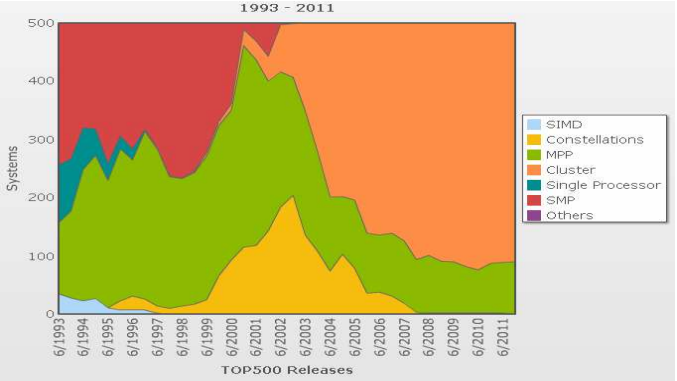
	Cost.	Computer	sito	Ncores	Rete conn	Linpack bench. (Pflops)
1	Fujitsu	K-computer	RIKEN AICS (Giappone)	705024 (proc SPARC64 VIIIfx 2.0 GHz)	Tofu interconnect	10.5
2	NUDT	Tianhe-1A	NUDT Tianjin (Cina)	186368 (proc Intel 6 core 2.93 GHz + Nvidia 2050 GPU)	custom	2.57
3	CRAY	Jaguar	ORNL (USA)	224162 (proc AMD 6 core 2.6 GHz)	custom	1.76
4	Dawning	Nebulae	NSCS (Cina)	120640 (proc Intel 6 core 2.67 GHz + Nvidia 2050 GPU)	Infiniband	1.27
5	HP	Tsubame 2/0	TiTech (Giappone)	73278 (proc Intel 6 core 2.93 GHz + Nvidia GPU)	Infiniband	1.19

Uso dei supercomputer



- La meta' dei supercomputers nella Top500 list e' usato nell'industria !!

Come e' fatto un supercomputer

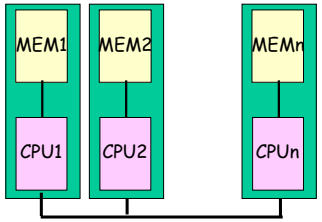


- L'80% dei supercomputer nella Top500 list sono cluster

Un cluster

- Un cluster e' un gruppo di computer connessi da una rete dedicata o switch ad alta velocita', che lavorano allo stesso problema, dando l'impressione di una unica risorsa di calcolo
- Ogni nodo ha una sua propria memoria e comunica con gli altri nodi attraverso espliciti scambi di messaggi
- Esistono cluster con migliaia di nodi

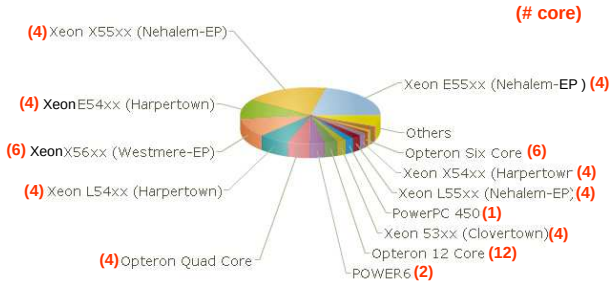
E' un esempio di calcolatore parallelo a memoria distribuita



Problemi dei cluster

- Comunicazione tra i nodi:
  - E' necessario trasferire dati in maniera esplicita con apposite librerie (MPI, PVM,..)
- Comunicazione
  - Rischio di nodi inattivi in attesa di messaggi
- Rete o switch:
  - i nodi sono molto piu' veloci delle reti:
  - Esempio Intel Woodcrest
    - A 3 GHz i 2 core possono eseguire 24 op.f.p. in un nanosecondo
    - Una rete Infiniband a 120 Gbit/sec puo' trasferire solo 4 dati reali tra i nodi in un nanosecondo

Come e' fatto un supercomputer



- Oltre il 90% dei supercomputer nella Top500 list utilizza CPU multicore

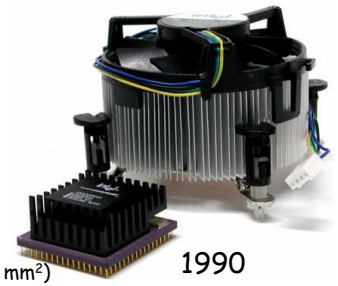
Perche' le CPU multicore

I processi industriali producono chip sempre piu' densi in termini di transistor

Inoltre:
Watt ~ Volt^2 x frequenza
Frequenza ~ Volt

Watt ~ frequenza^3

Il rapporto Watt/mm^2 (calore per mm^2) diventa troppo grande per raffreddare i chip economicamente



Perche' le CPU multicore

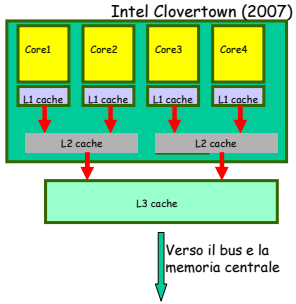
	core	freq.	perf.	potenza
CPU1	1	1	1	1
CPU2	1	1.5	1.5	3.3
CPU3	2	0.75	1.5	0.8

CPU2 con 2 core e una frequenza ridotta del 25%, ha le stesse prestazioni di CPU3 con 1/3 della potenza (e del calore dissipato)

Una CPU multicore

- Una CPU multicore e' un insieme di unita' processanti autonome in un unico chip, che condividono risorse (cache, bus, e memoria centrale
- I core possono condividere la cache o avere cache proprie
- Oggi si hanno al piu' 6-8 core

E' un esempio di calcolatore parallelo a memoria condivisa



Problemi delle CPU multicore

- Comunicazione tra i core:
  - Non c'e' comunicazione diretta ma solo attraverso le cache o la memoria centrale (risorse condivise)
- Sincronizzazione
  - Rischio di core inattivi
- Bus verso la memoria:
  - i core sono molto piu' veloci della memoria:
  - Esempio Intel Woodcrest
    - A 3 GHz i 2 core possono eseguire 24 op.f.p. in un nanosecondo
    - A 10.5 GB/sec il bus puo' trasferire solo 2.5 dati reali alla CPU in un nanosecondo
    - Essenziale un uso efficiente delle cache

Come e' fatto un supercomputer

Rank	Site	Computer
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT
3	DOE/SC/Oak Ridge National Laboratory United States	Cray XT5-HE Opteron 6-core 2.6 GHz Cray Inc.
4	National Supercomputing Centre in Shenzhen (NSCS) China	Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 Dawning
5	GSIC Center, Tokyo Institute of Technology Japan	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows NEC/HP
6	DOE/NNSA/LANL/SNL United States	Cray XE6, Opteron 6136 8C 2.40GHz, Custom Cray Inc.
7	NASA/Ames Research Center/NAS United States	SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon 5570/5670 2.93 Ghz, Infiniband SGI
8	DOE/SC/LBNL/NERSC United States	Cray XE6, Opteron 6172 12C 2.10GHz, Custom Cray Inc.
9	Commissariat a l'Energie Atomique (CEA) France	Bull bullx super-node S6010/S6030 Bull
10	DOE/NNSA/LANL United States	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband IBM

- Tra i primi 10 supercomputer della Top500 list, 3 utilizzano acceleratori basati su GPU (Graphic Processing Unit)

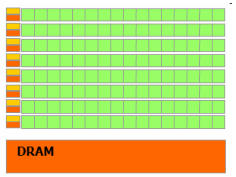
Le GPU

- Le GPU sono nate come processori specializzate per la grafica 3D e si sono sviluppate sotto la spinta delle esigenze dei videogames
- Esempio: un cubo in movimento

Stesse operazioni su tutti i vertici del cubo

Composte da centinaia di unita' processanti elementari capaci di eseguire in parallelo la stessa operazione

E' un esempio di calcolatore parallelo di tipo SIMD



Problemi delle GPU

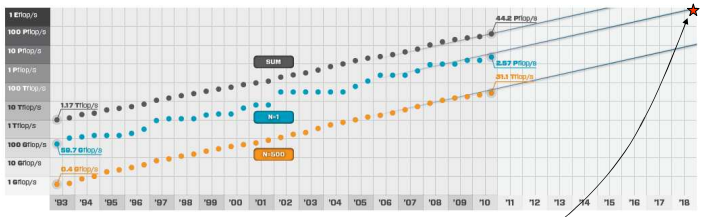
- Efficiente in casi particolari dove e' necessario eseguire le stesse operazioni su molti dati
  - Applicazioni data parallel
  - ad es. Algebra lineare numerica
- Interfacciamento con la memoria
  - Es. Nvidia Tesla C1060
    - Peak performance 933 Gflop/sec s.p.
    - Memory bandwidth 102 GB/sec (25×10^9 dati reali/sec)
- Necessari ambienti software che tengono conto del particolare hardware

Tianhe 1A: #2 nella Top500 list

- Cluster di 7168 nodi di calcolo in 112 rack
- Ogni nodo composto da
  - 2 CPU Intel Xeon 6-core
  - 1 GPU Nvidia

3 distinte metodologie di programmazione

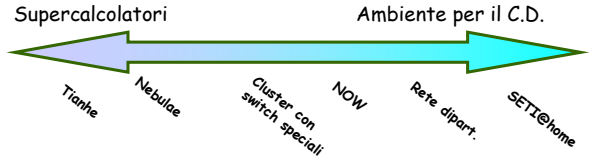
Uno sguardo al futuro



Proiezione dei dati della Top500 list

- Sistema da 1 Esaflops
  - Previsto nel 2018-2019
  - Circa 10^7 core (10^9 threads)
  - Architettura ibrida multicore/GPU integrata
  - Nuovi aspetti critici: energia e fault tolerance

Non solo supercalcolatori



- reti veloci
- risorse limitate
- risorse dedicate e omogenee
- applicazione gestisce le risorse
- costo hardware notevole
- overhead sw sistema < 5%
- presenza di vincoli temporali
- es. Tianhe 1A
  - 186638 core
  - 2.5 Pflops

- reti lente
- risorse potenzialmente illimitate
- risorse condivise e disomogenee
- ambiente sw.gestisce le risorse
- costo hardware trascurabile
- overhead sw sistema > 20%
- assenza di vincoli temporali
- es. SETI@home
  - 5 milioni CPU
  - 769 Tflops

Esempio : Progetto SETI

- Progetto per la ricerca di intelligenze extraterrestri
- Dati raccolti dal radiotelescopio di Arecibo
- Trovare sequenze regolari nelle frequenze all'interno dei segnali raccolti

- Migliaia di nastri da 35 GB
- ogni nastro da 35 Gbyte e' diviso in oltre 150000 workunit di circa 350 Kb

Le dimensioni del progetto

- ogni workunit contiene i segnali raccolti in circa 100 sec

- Le workunit vengono spedite ai client partecipanti ed inattivi che eseguono FFT con diversi campionamenti ed analisi statistiche per circa $3x10^{12}$ flops (circa 15 ore su PC con proc 3 GHz)

- i risultati vengono rispediti ai server ad Arecibo

- Piu' grande progetto di calcolo distribuito (769 Tflops)

- Numerosi progetti analoghi
  - genome@home
  - folding@home (oltre 5 Pflops !!)
  - QMC@home
  - BOINC

E' un esempio di sistema distribuito

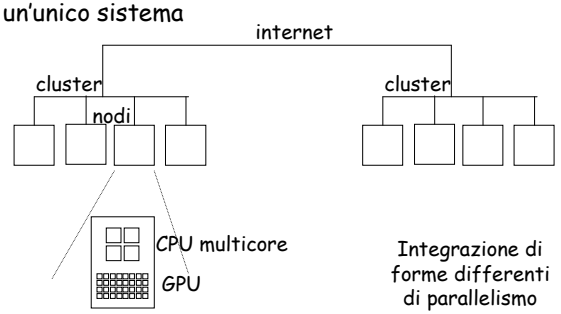
Calcolo parallelo e distribuito

- GPU computing
- Multicore computing (c.p. a memoria condivisa)
- Cluster computing (c.p. a memoria distribuita)
- Distributed computing

Aspetti differenti del calcolo parallelo che richiedono differenti metodologie per lo sviluppo di algoritmi

Parallelismo gerarchico

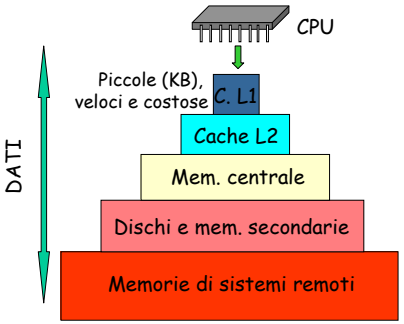
- Sempre piu' spesso tali forme coesistono in un'unico sistema



Integrazione di forme differenti di parallelismo

Problema principale : movimento dei dati (tra sistemi e tra memorie)

La memoria gerarchica



Fondamentale ridurre le comunicazioni e gli accessi alle memorie

Grandi (TB), lente e economiche

Bandwidth e Latenza

- Bandwidth e Latenza sono una misura del costo per muovere i dati
- Esempio:
  - Bandwidth: quanti auto/ora possono andare tra Napoli e Roma?
    - #auto/ora = densita' (#auto/km) × velocita' (km/ora) × #corsie
  - Latenza: quanto impiega la prima auto per andare da Napoli a Roma?
    - #ore = distanza (#km) / velocita' (km/ora)
- Esempio: sistema con 2 CPU
  - Bandwidth: quanti bit/sec possono essere trasferiti tra le CPU?
    - #bit/sec = densita' (#MByte/cm) × velocita' (cm/sec) × #collegamenti
  - Latenza: quanto impiega il primo bit ad arrivare?
    - #sec = distanza (cm) / velocita' (cm/sec)
- Per leggere dati dal disco alla memoria – stessa idea
- Per spostare dati dalla memoria alle cache – stessa idea

Bandwidth e Latenza

- Notazioni:
  - #Byte come unita' di misura
  - 1/Bandwith ≡ β, tempo in sec per trasferite un Byte
  - Latenza ≡ α, tempo in sec per far arrivare il primo Byte
- Modello di comunicazione
  - Per spostare un messaggio di n Byte occorrono:

$$t = \alpha + \beta\, n \quad sec$$

- Il tempo di esecuzione di un algoritmo e' la somma di:

T_{flop} = # flop * t_{flop}

T_{com} e T_{mem} = { # Byte comunicati / bandwidth ; # messaggi * latenza

Evitare le comunicazioni

- # flops * t_{flop}
- # Bytes / bandwidth
- # messaggi * latenza

} Comunicazioni

- t_{flop} << 1/ bandwidth << latency
  - La differenza crese con il tempo !!

Crescita annuale delle prestazioni			
t_{flop}		Bandwidth	Latenza
60%	Rete	26%	15%
	RAM	23%	7%

- Obiettivo : riorganizzare gli algoritmi in amniera da evitare tutte comunicazioni tra tutti i livelli della memoria gerarchica

  L1 ⟷ L2⟷ RAM ⟷ dischi ⟷ reti, etc

33

Esempi

- Il movimento dei dati e' molto piu' costoso delle operazioni aritmetiche
- Riduzione della comunicazione all'interno di un singolo sistema
  - Registri CPU ↔ L1 ↔ L2 ↔ RAM ↔ dischi
- Riduzione della comunicazione tra unita' di calcolo
  - Tra core in una CPU multicore
  - Tra CPU in un singolo sistema (es. CPU e GPU)
  - Tra schede in un rack
  - Tra rack in un supercomputer
  - Tra citta' in un sistema distribuito
- Come possiamo fare?
  - Non possiamo sperare nei sistemi operativi e nei compilatori
  - Occorrono nuovi algoritmi

34

Obiettivi del corso

- Esplorare varie forme di parallelismo ed acquisire metodologie per lo sviluppo di algoritmi efficienti in differenti ambienti computazionali
  - Sistemi paralleli a memoria distribuita (cluster)
  - Sistemi paralleli a memoria condivisa (proc. multicore)
  - Sistemi ibridi con acceleratori f.p. (GPU)
  - Sistemi distribuiti
- Mediante l'utilizzo di alcuni case study
  - Prodotto di matrici
  - Algoritmi di fattorizzazione dell'algebra lineare

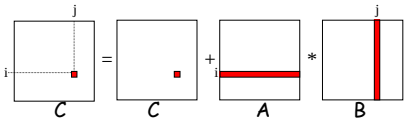
Obiettivi del corso

Comprendere in maniera critica

- Quando il calcolo parallelo e' utile
- Le caratteristiche degli ambienti (hw e sw) di programmazione parallela.
- Algoritmi che tengano conto dell'hardware e dei modelli
- Alcune questioni legate al calcolo ad alte prestazioni (HPC)

I nostri casi di studio nel corso

- Operazione su matrici $C = \beta C + \alpha AB$ di ordine n



$$C(i,j) = \beta C(i,j) + \alpha \sum_{k=1}^{n} A(i,k) B(k,j)$$
$$i,j = 1,\ldots,n$$

- Un importante nucleo computazionale in molti problemi
  - Utilizzato in tutti i problemi di algebra lineare (fattorizzazioni, autovalori, metodi iterativi,…)
- E' possibile utilizzare le idee sviluppate in altri contesti
- Interessante rapporto tra dati $O(n^2)$ e operazioni $O(n^3)$
- Uno degli algoritmi piu' studiati in assoluto

osservazione

- Per qualunque combinazione di indici

```
for __ =1 to n
        for __ =1 to n
                for __ =1 to n
                        C(i,j) = βC(i,j) + αA(i,k)B(k,j)
                endfor
        endfor
endfor
```

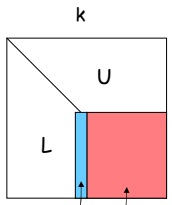
- Stessa complessita' computazionale
- Si ottiene sempre lo stesso risultato !!

Algoritmo di Gauss

- Piu' nota fattorizzazione
- A deve essere non singolare
- A = LU (L triangolare inf, U triangolare sup)
- Algoritmo instabile
  - Necessario pivoting
- Varie versioni
  - "classica" (forma kji)
  - Crout, Doolittle

Algoritmo di Gauss

```
.....
for k = 1 to n-1
    max=A(k,k)      L=k                          pivoting
    for i = k+1 to n
            if (A(i,k) > max)
                    max=A(i,k)
                    L=i
            endif
    endfor
    for j = k to n
            t = A(L,j)
            A(L,j) = A(k,j)
            A(k,j) = t
    endfor
    for i = k+1 to n
            A(i,k) = A(i,k) / A(k,k)
    endfor
    for j = k+1 to n
            for i = k+1 to n
                A(i,j) = A(i,j) – A(i,k)*A(k,j)
            endfor
    endfor
endfor
```



Calcolo moltiplicatori

Aggiornamento matrice attiva

Algoritmo di Cholesky

- Classico esempio di fattorizzazione
- Non fa uso del pivoting

- $A = LL^T$ (L triangolare inferiore)
- A deve essere
  - Simmetrica ($A^T = A$)
  - Definita positiva ($A^TAX > 0$ per ogni x non nullo)

- Teorema:
  - A simmetrica a diagonale dominante ➔ A definita positiva

- Varie versioni

Algoritmo right looking

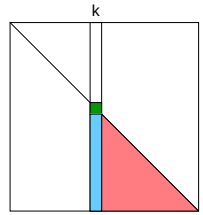
```
do k = 1,N

  A(k,k) = sqrt(A(k,k))

  do i = k+1,N
    A(i,k) = A(i,k)/A(k,k)
  enddo

  do j = k+1,N
     do i = j,N
       A(i,j) = A(i,j)-A(i,k)*A(j,k)
     enddo
  enddo

enddo
```



Aggiorna le colonne a destra della colonna k

Algoritmo left looking

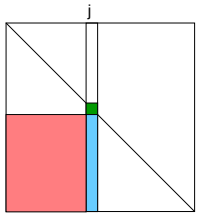
```
do j = 1,N

   do k = 1,j-1
      do i = j,N
        A(i,j) = A(i,j)-A(i,k)*A(j,k)
      enddo
   enddo

   A(j,j) = sqrt(A(j,j))

   do i = j+1,N
      A(i,j) = A(i,j)/A(j,j)
   enddo

enddo
```



Aggiorna la colonna j utilizzando le colonne a sinistra