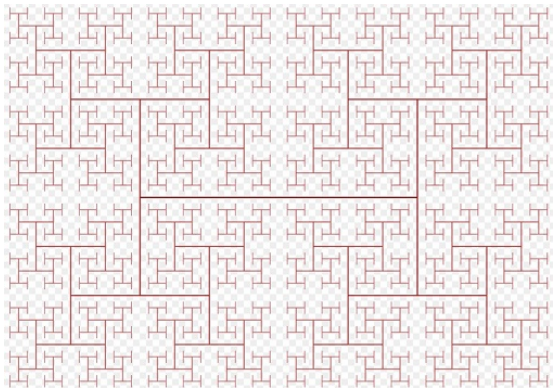
Quarenta+ anos de Computação Paralela - uma visão pessoal

Siang Wun Song

<song@ime.usp.br>

2.o Encontro Paulista de Pós-Graduandos em Computação
Campinas, 8 de novembro de 2018

Slides em https://www.ime.usp.br/∼song



Computação Paralela de 1976 - 2018

- Em 1976 - iniciei o meu programa de doutorado na área de Computação Paralela na Carnegie Mellon University.
- Darei aqui uma visão pessoal sobre a evolução e as fases mais marcantes da computação paralela dos últimos 42 anos.
- Vamos então fazer uma viagem ao passado, voltando 40+ anos ...

Computação Paralela de 1976 - 2018

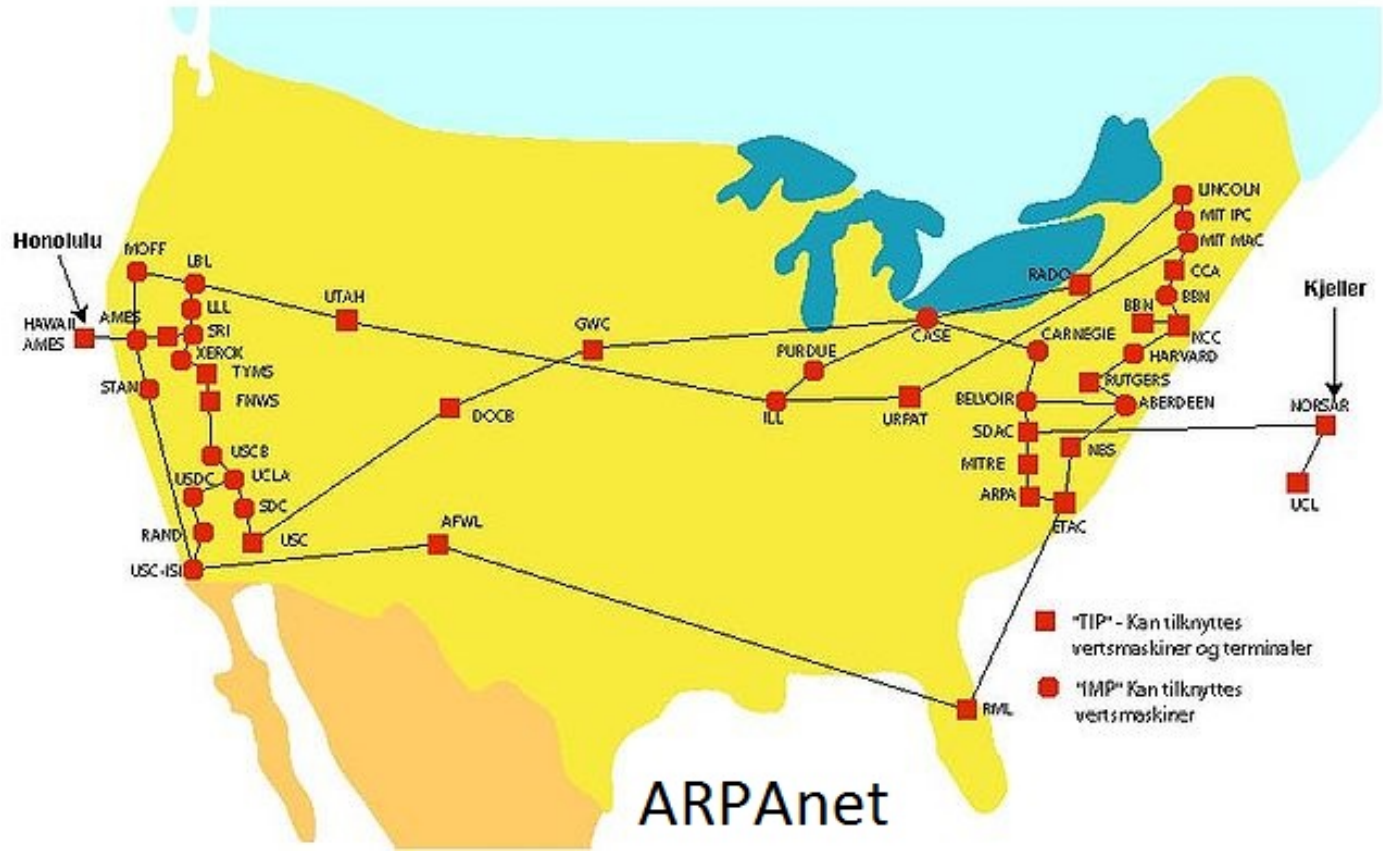
- Em 1976 - iniciei o meu programa de doutorado na área de Computação Paralela na Carnegie Mellon University.
- Darei aqui uma visão pessoal sobre a evolução e as fases mais marcantes da computação paralela dos últimos 42 anos.
- Vamos então fazer uma viagem ao passado, voltando 40+ anos ...

Início do doutorado na CMU em 1976

CMU Dept of Computer Science 1976

Em 1976 iniciei meu doutorado em Ciência da Computação na Carnegie Mellon University. Hoje o depto. se tornou CMU School of Computer Science - no Prédio Gates-Hillman.

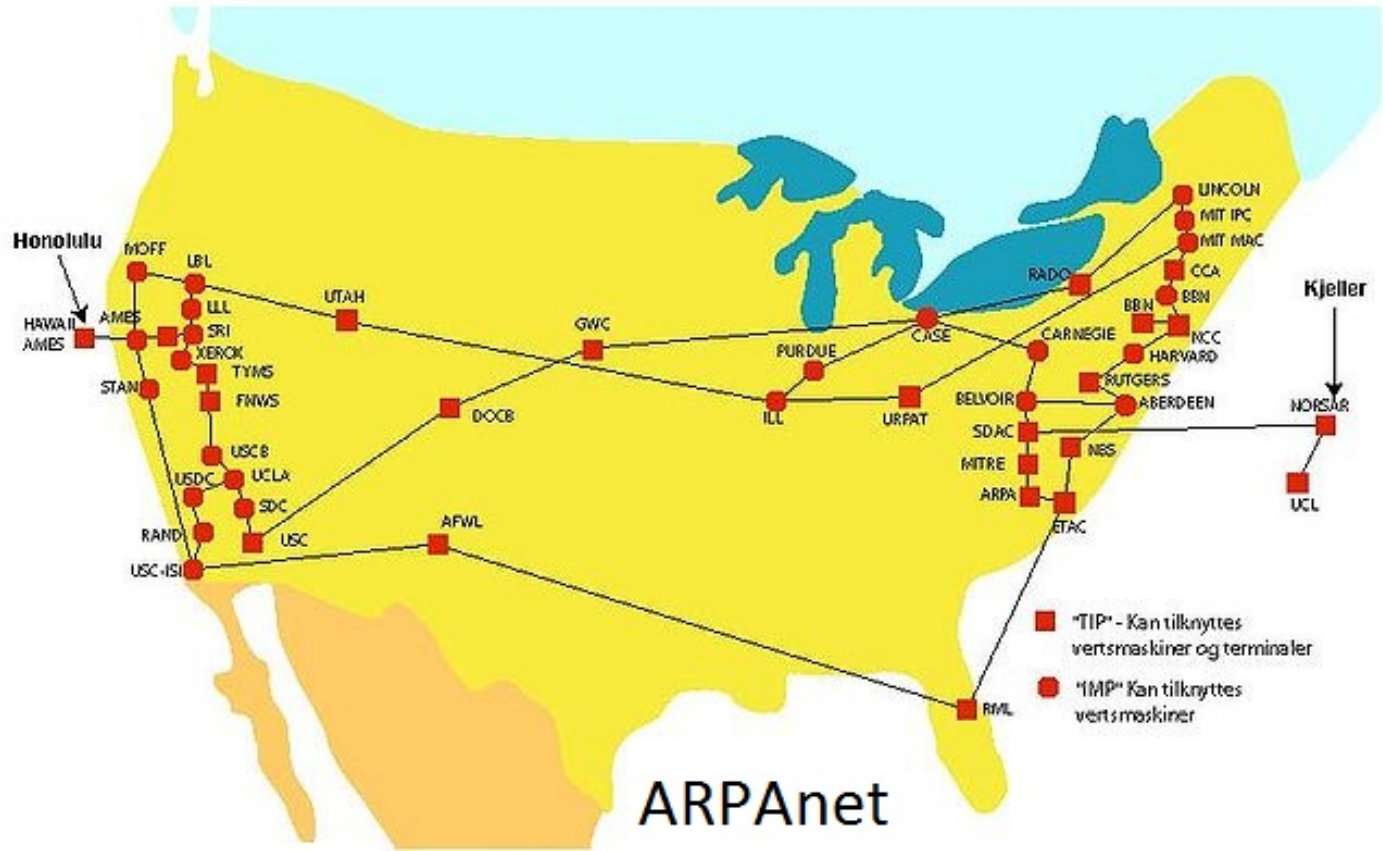
Início do doutorado na CMU em 1976



Em 1976, a Carnegie Mellon University fazia parte da ARPAnet, que conectava algumas universidades e laboratórios de pesquisa. Mas não havia WWW, nem facebook, instagram, Google, Wikipedia, ...

Vocês são felizes e não sabiam

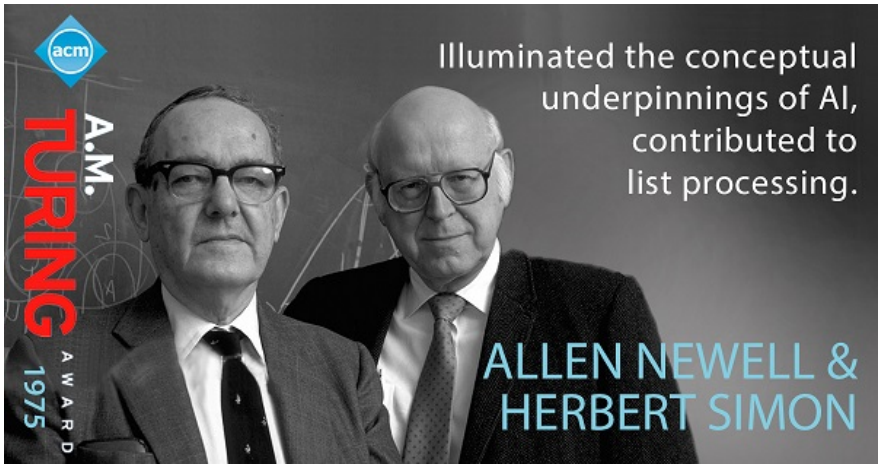
Início do doutorado na CMU em 1976



Em 1976, a Carnegie Mellon University fazia parte da ARPAnet, que conectava algumas universidades e laboratórios de pesquisa. Mas não havia WWW, nem facebook, instagram, Google, Wikipedia, ...

Vocês são felizes e não sabiam ...

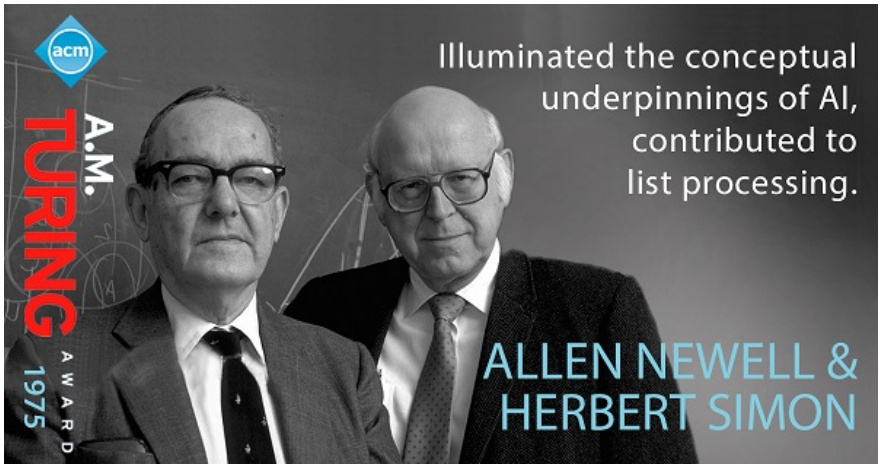
Escolha do orientador e tema de pesquisa



- A área pretendida era Inteligência Artificial. Allen Newell e Herbert Simon receberm o Turing Award em 1975. Herbert Simon recebu ainda o Prêmio Nobel em Economia em 1978.
- Porém o projeto vigente na área era Reconhecimento da Fala em inglês, o que pouco me interessou.
- Acabei escolhendo a Computação Paralela, uma área surgindo no departamento.

PRIMEIRA LIÇÃO: É necessário casar o interesse do aluno com o do orientador e com os projetos em andamento na instituição.

Escolha do orientador e tema de pesquisa



- A área pretendida era Inteligência Artificial. Allen Newell e Herbert Simon receberm o Turing Award em 1975. Herbert Simon recebu ainda o Prêmio Nobel em Economia em 1978.
- Porém o projeto vigente na área era Reconhecimento da Fala em inglês, o que pouco me interessou.
- Acabei escolhendo a Computação Paralela, uma área surgindo no departamento.

PRIMEIRA LIÇÃO: É necessário casar o interesse do aluno com o do orientador e com os projetos em andamento na instituição.

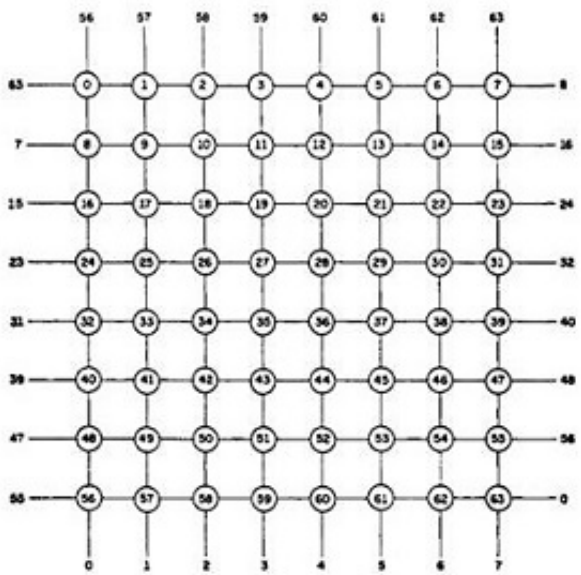
Computação paralela em 1976?

- Computação paralela precisa de computador paralelo.
- Quantos computadores paralelos havia na época?
- Hmmmm, deixe-me ver ...
- Ah, havia uns dois ou três ...

Computação paralela em 1976?

- Computação paralela precisa de computador paralelo.
- Quantos computadores paralelos havia na época?
- Hmmmm, deixe-me ver ...
- Ah, havia uns dois ou três ...

Computador paralelo ILLIAC IV 1975

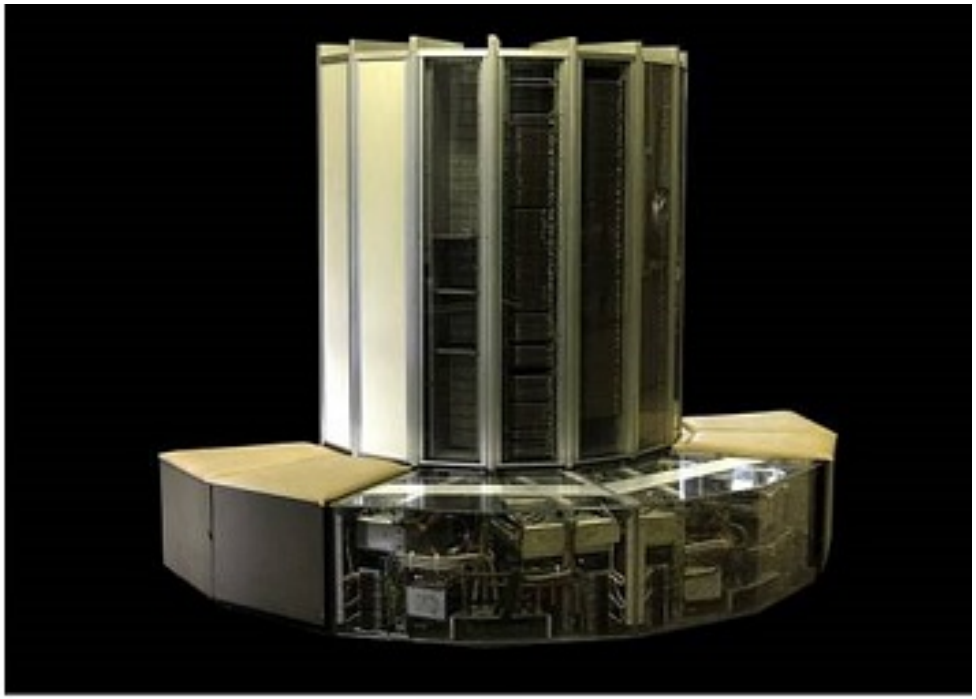


Source: Wikipedia

- Illiac IV (Univ. of Illinois - fabricada por Burroughs) 1975
- Devido a orçamento, a máquina construída tinha uma CPU controlando 64 unidades de ponto flutuante. (O projeto original previa 4 CPUs e 256 unidades de ponto flutuante.)
- Todas as unidades de ponto flutuante executam a mesma instrução (modelo SIMD - *Single Instruction Multiple Data*).

Computador paralelo Cray 1 - 1976

Seymour Cray



Cray 1 - ETH - Zurich

- Cray-1 (Cray Research) 1976
- Possui instruções vetoriais adequadas para processamento matricial.
- Apresenta 12 unidades funcionais implementadas com a técnica de *pipelining*.

Computador paralelo C.mmp - 1977

William Wulf

- C.mmp (Willam Wulf - Carnegie Mellon University) 1977

  W. A. Wulf and C. G. Bell. C.mmp - A multi-mini-processor. *Fall Joint Comp. Conf.*, 1972, pp. 765-777.

- Composto por 16 minicomputadores PDP-11 e 16 bancos de memória de 128 Kbytes, conectados por um *cross-bar*.
- Modelo MIMD (*Multiple Instruction Multiple Data*).

Edsger Dijkstra e projeto de algoritmos paralelos

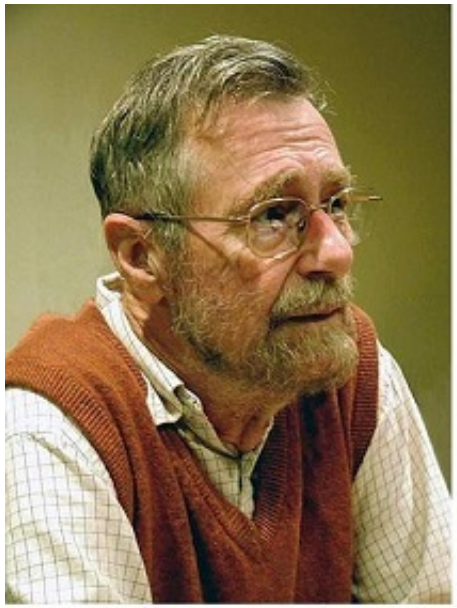
Edsger W. Dijkstra

Dining philosophers

Na falta de computadores paralelos, a moda era projetar algoritmos paralelos no papel e demonstrar a sua corretude.

- Edsger W. Dijkstra (Turing Award 1972) publicou vários trabalhos sobre processos cooperantes.
- Além do clássico problema de *Dining Philosophers*, um artigo recebeu considerável atenção: *On-the-fly garbage collection*.

On-the-fly garbage collection



Operating Systems — R.S. Gaines, Editor

On-the-Fly Garbage Collection: An Exercise in Cooperation

Edsger W. Dijkstra
Burroughs Corporation

Turing Award 1972

Leslie Lamport
SRI International

Turing Award 2013

A.J. Martin, C.S. Scholten, and E.F.M. Steffens
Philips Research Laboratories

Edsger W. Dijkstra, Leslie Lamport, A. J. Martin, C. S. Scolten and E. F. M. Steffens. On-the-fly garbage collection: an exercise in cooperation. In: Language Hierarchies and Interfaces, F. L. Bauer and K. Samalson (editors), Springer-Verlag, 1976, pp. 43-55.

- Ao mesmo tempo em que um programa LISP constrói listas a partir do espaço livre, um outro processo já tenta identificar e coletar *lixo*.

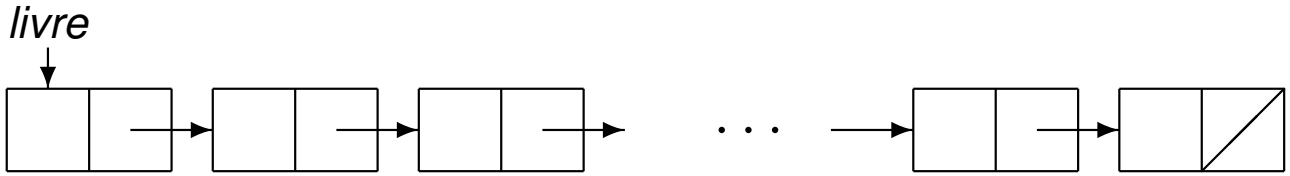
Algoritmo paralelo para *garbage collection*

- Coleção de lixo é o nome de uma rotina básica de alguns sistemas (por exemplo, LISP, Java, sistemas operacionais baseados em listas de capacidades, etc.) para recuperar células de memória que haviam sido alocadas para uso por alguma aplicação mas posteriormente ficaram em desuso pela aplicação, sem no entanto terem sido retornadas ao espaço disponível do sistema.

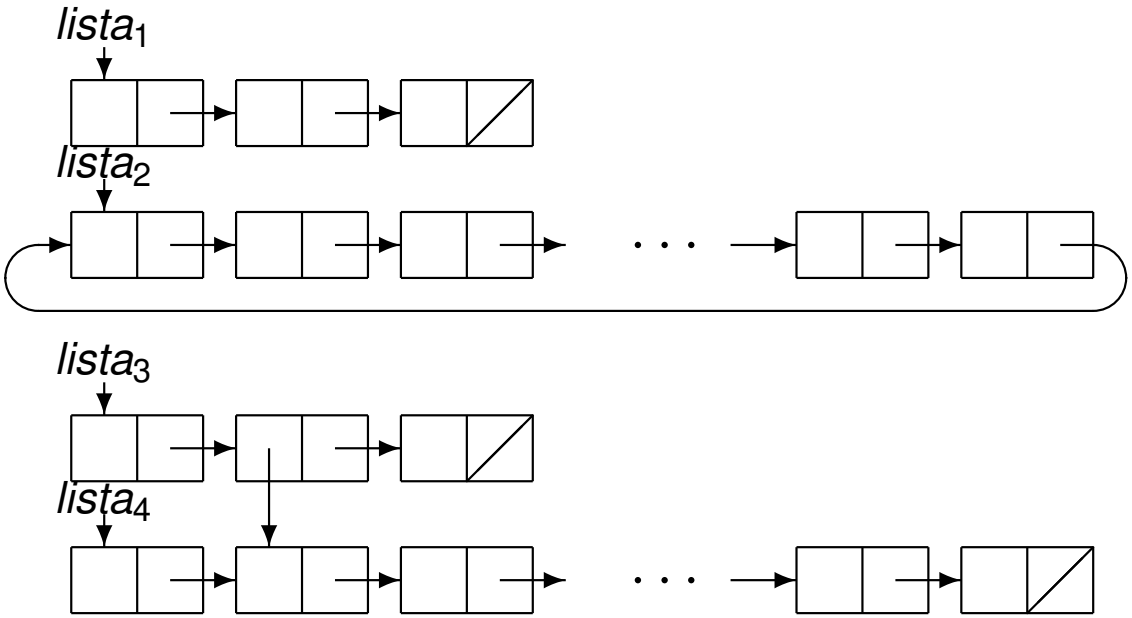
- É papel dessa rotina identificar e reaproveitar tais células de memória.

Um esquema de alocação dinâmica de memória

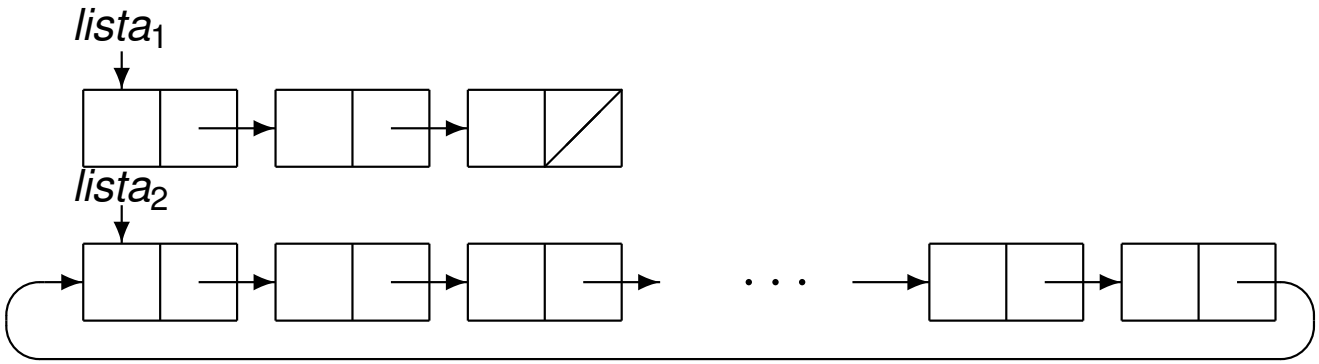
- O espaço livre organizado inicialmente como uma longa lista livre.



- Células livres são retiradas da lista livre para serem usadas na construção de estruturas de dados de aplicações.

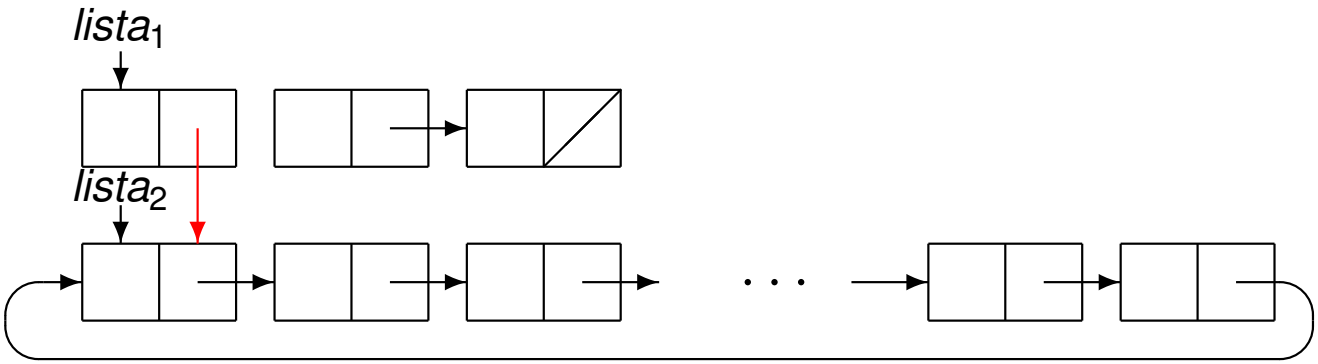


Exemplo do surgimento de lixo



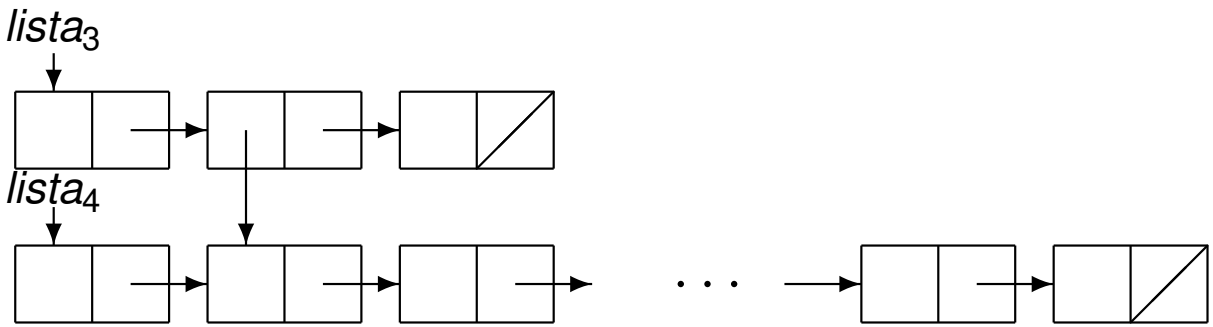
- Um ponteiro do primeiro elemento da $lista_1$ foi mudado de tal forma que a segunda unidade de espaço em diante não é mais acessível.

Exemplo do surgimento de lixo



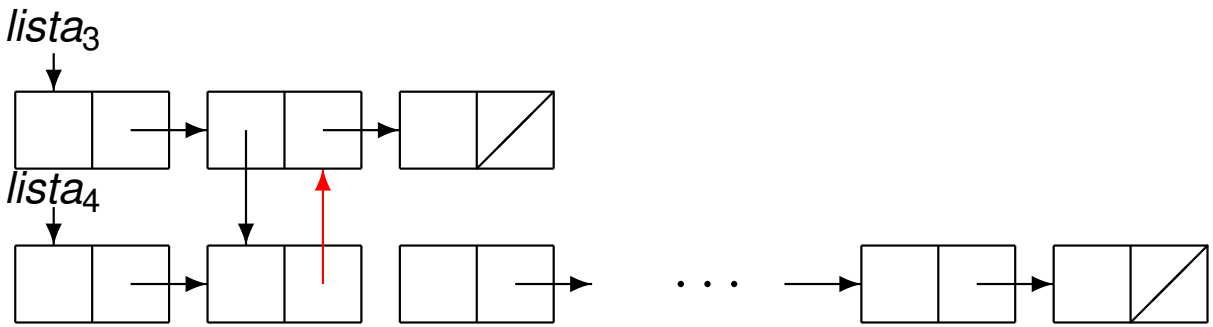
- Um ponteiro do primeiro elemento da $lista_1$ foi mudado de tal forma que a segunda unidade de espaço em diante não é mais acessível.

Exemplo do surgimento de lixo



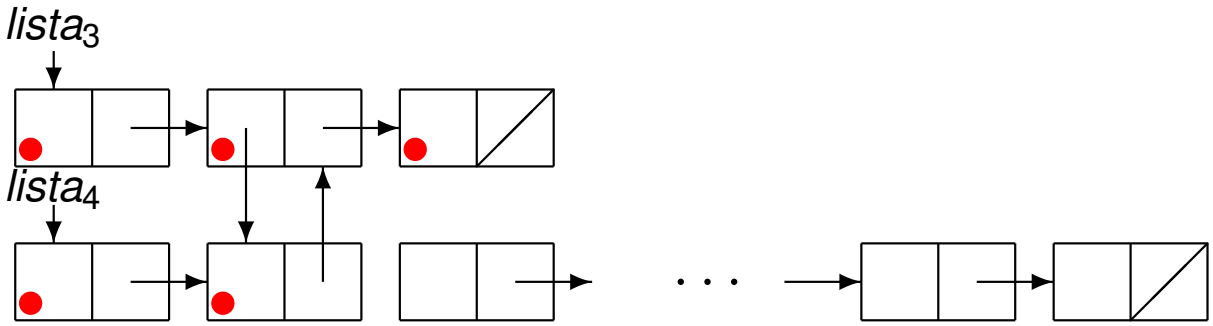
- Um outro exemplo.

Exemplo do surgimento de lixo



- Um outro exemplo.

Coletor de lixo - Método marca-varre (*mark-scan*)



- Usa um bit adicional por célula para servir de marca (●). Consta de duas fases.
- Fase de marcação (*mark*): com a ajuda de uma pilha, marca todas as células acessíveis a partir de ponteiros externos conhecidos.
- Fase de varredura (*scan*): Visita cada célula sequencialmente. Se marcada (●), então apaga marca e segue em frente. Se não marcada então recolhe para reuso.

On-the-fly garbage collection

Operating Systems — R.S. Gaines, Editor

On-the-Fly Garbage Collection: An Exercise in Cooperation

Edsger W. Dijkstra
Burroughs Corporation

Turing Award 1972

Leslie Lamport
SRI International

Turing Award 2013

A.J. Martin, C.S. Scholten, and E.F.M. Steffens
Philips Research Laboratories

- Dijkstra propõe usar um processador para executar o programa LISP e outro para concorrentemente identificar e coletar lixo.
- O coletor de lixo usa uma pilha para implementar a fase de marcação.
- Dijkstra observa que quando um programa LISP muda um ponteiro de um lugar para uma posição x, então a posição x também precisa entrar na pilha de marcação.
- Como a pilha é compartilhada, a solução de Dijkstra usa uma região crítica quando se manipula a pilha. Cada célula de memória possui uma marca de 2 bits (codificando uma de 3 cores).

Outros algoritmos paralelos para coleta de lixo

Dois outros artigos, também sobre coleta de lixo concorrente, causaram bastante interesse na época e foram premiados.

- Guy L. Steele Jr. Multiprocessing compactifying garbage collection. *Communications of the ACM*, Volume 18, No. 9, September 1975 pp. 495-508. First place 1975 ACM Student Paper Award.
- Philip L. Wadler. Analysis of an algorithm for real time garbage collection *Communications of the ACM*, Vol. 19, No. 9, September 1976, pp. 491-500. First place 1976 ACM Student Paper Award.

Notem que ambos os artigos foram de alunos de pós-graduação.

Algoritmos paralelo de Kung e Song 1977

- O algoritmo paralelo para coleta de lixo concorrente de Kung e Song apresentado na FOCS 1977:

  H. T. Kung and S. W. Song. An efficient garbage collection system and its correctness proof. 18th IEEE Annual Symposium on Foundations of Computer Science (FOCS). 1977, pp. 120-133.

- Não requer uso de região crítica e não usa nenhuma operação primitiva de sincronização. Apenas requer que um acesso à memória seja uma operação atômica (indivisível). Cada célula de memória é marcada com uma de 4 cores (2 bits).
- O trabalho foi apresentado no segundo ano de doutorado. Este foi um dos trabalhos mais importantes e mais citados.

SEGUNDA LIÇÂO: Tendo pesquisado e estudado muito um assunto, o estudante de pós-graduação pode produzir pesquisas de boa qualidade, como um pesquisador experiente. É preciso ter confiança da própria capacidade.

Algoritmos paralelo de Kung e Song 1977

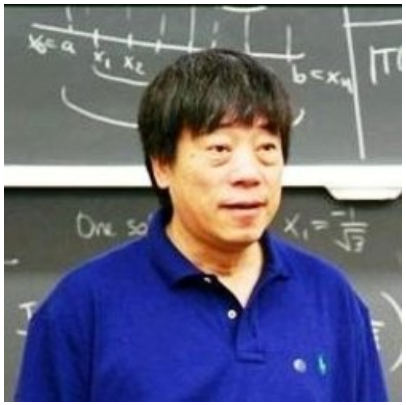
- O algoritmo paralelo para coleta de lixo concorrente de Kung e Song apresentado na FOCS 1977:

  H. T. Kung and S. W. Song. An efficient garbage collection system and its correctness proof. 18th IEEE Annual Symposium on Foundations of Computer Science (FOCS). 1977, pp. 120-133.

- Não requer uso de região crítica e não usa nenhuma operação primitiva de sincronização. Apenas requer que um acesso à memória seja uma operação atômica (indivisível). Cada célula de memória é marcada com uma de 4 cores (2 bits).
- O trabalho foi apresentado no segundo ano de doutorado. Este foi um dos trabalhos mais importantes e mais citados.

SEGUNDA LIÇÂO: Tendo pesquisado e estudado muito um assunto, o estudante de pós-graduação pode produzir pesquisas de boa qualidade, como um pesquisador experiente. É preciso ter confiança da própria capacidade.

Systolic arrays 1978 - H. T. Kung



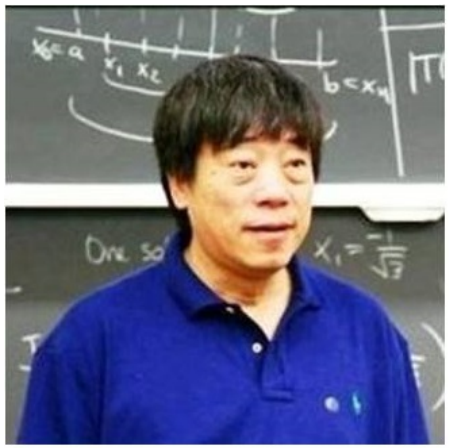
- O avanço da tecnologia de VLSI (circuito integrado em pastilha de silício), viabilizou a confecção de pastilhas sob medida para aplicações específicas.

  Lei de Moore: O número de transistores em uma pastilha de silício dobra a cada 18 meses.

- Em 1978, H. T. Kung inventou os chamados *systolic arrays*, que implementa algoritmos específicos diretamente na pastilha.

  H. T. Kung. Why systolic architectures? *Computer*, Vol. 15, No. 1, 1982, pp. 37-46.

Systolic arrays 1978



H. T. Kung

Charles Leiserson

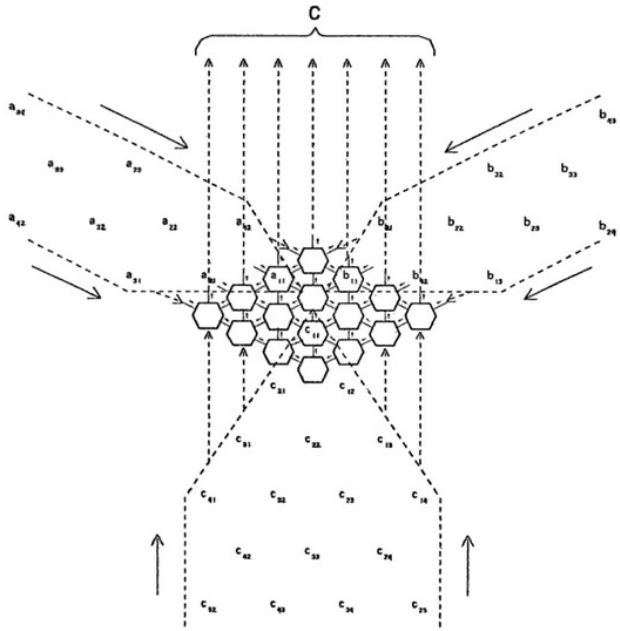


Figure 4-2: The hex-connected systolic array for the matrix multiplication problem in Figure 4-1.

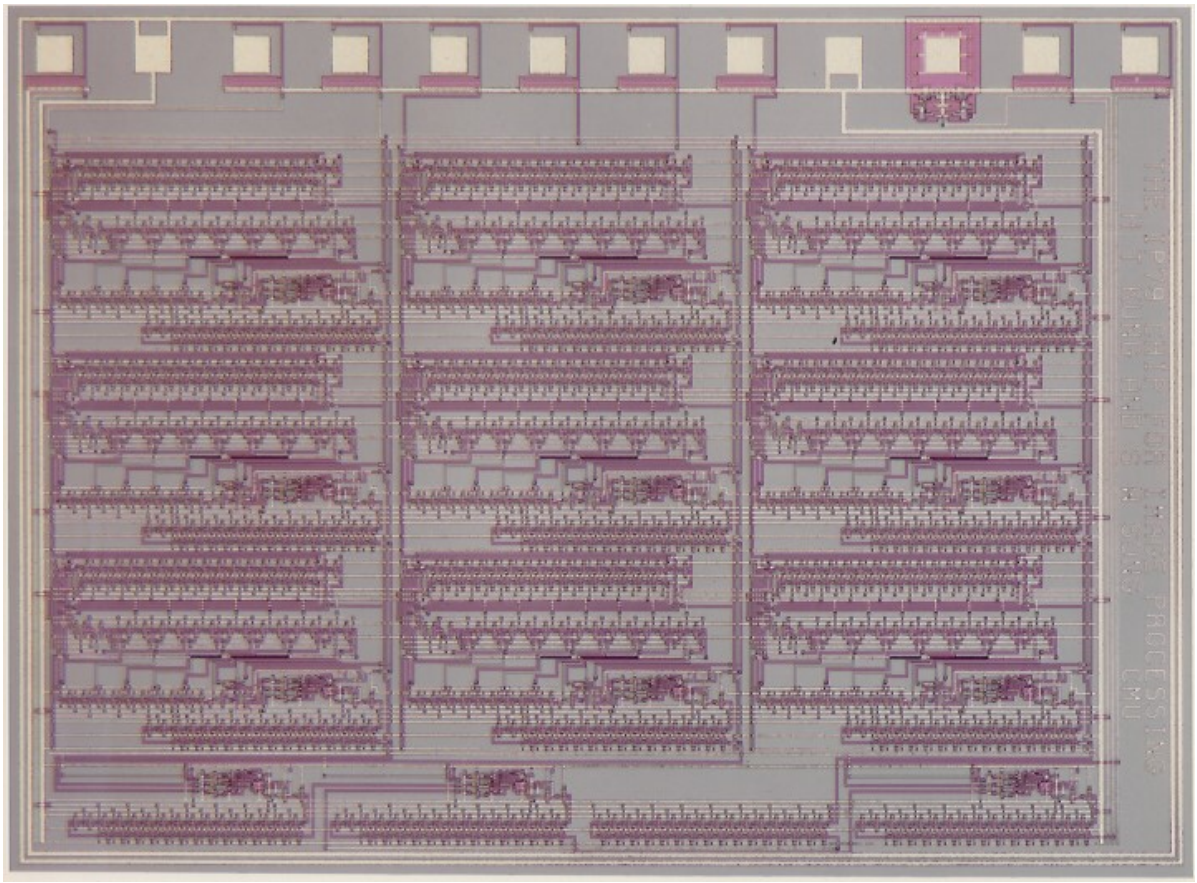
- Kung e Leiserson ficaram famosos com o projeto do primeiro *systolic array* para multiplicação de matrizes.
- Outros *systolic arrays* foram projetados para realizar: ordenação, transformada de Fourier, solução de sistemas lineares, etc, diretamente em hardware, em pastilhas de silício.

Um outro array sistólico para multiplicação de matrizes

Okuda, Kunio & Song, S. W. Um Algoritmo de Multiplicação de Matrizes para implementação em VLSI. Anais do I Congresso da Sociedade Brasileira de Microeletrônica. Campinas, julho de 1986, pp. 383-393.

Okuda e Song: Array sistólico para multiplicação de matrizes.

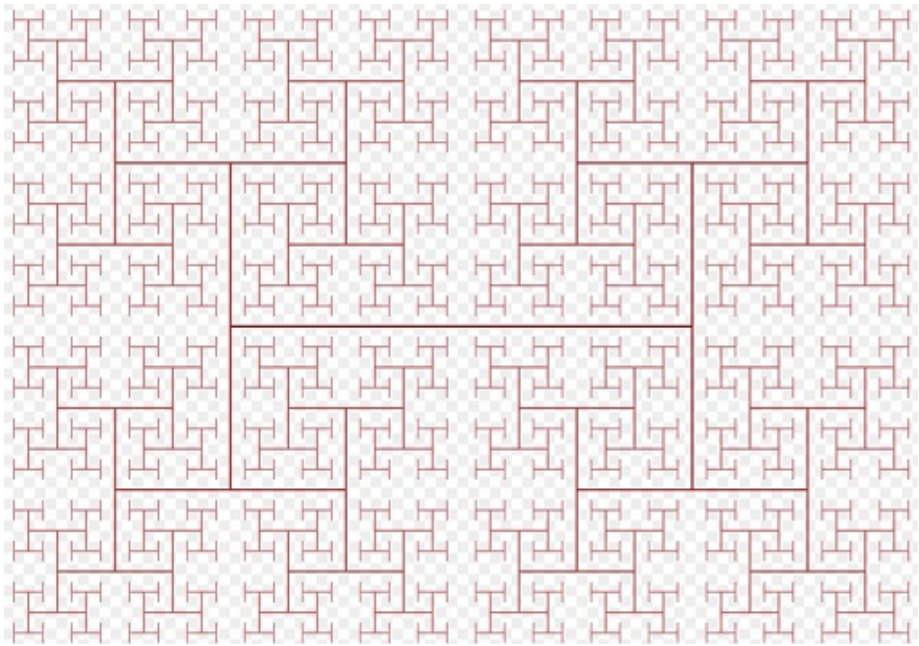
2D Convolution chip - Kung e Song



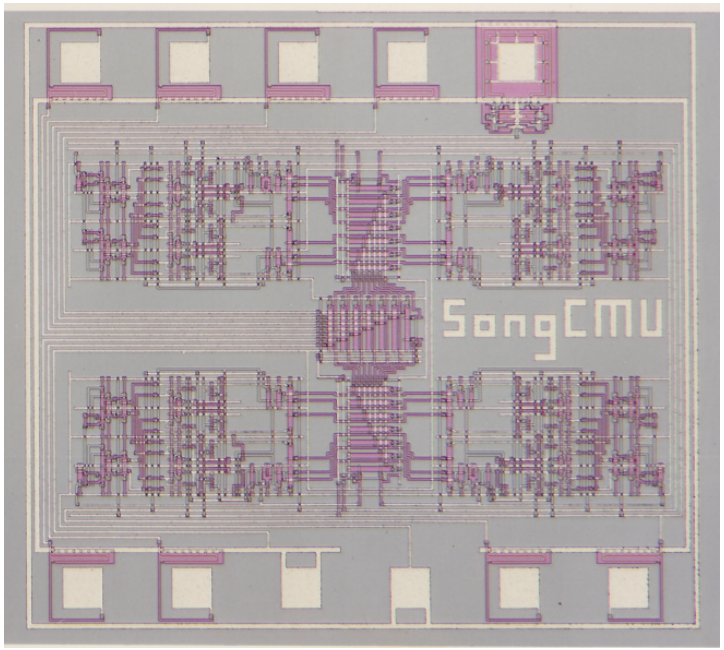
Um artigo que despertou bastante interesse na comunidade.

Kung, H. T. and Song, S. W. A Systolic 2-D Convolution Chip. In: Multi-Computers and Image Processing: Algorithms and Programs*, K. Preston and L. Uhr (editors), Academic Press, 1982, pp. 373-384.*

Outro array sistólico: arvore binária para fazer busca

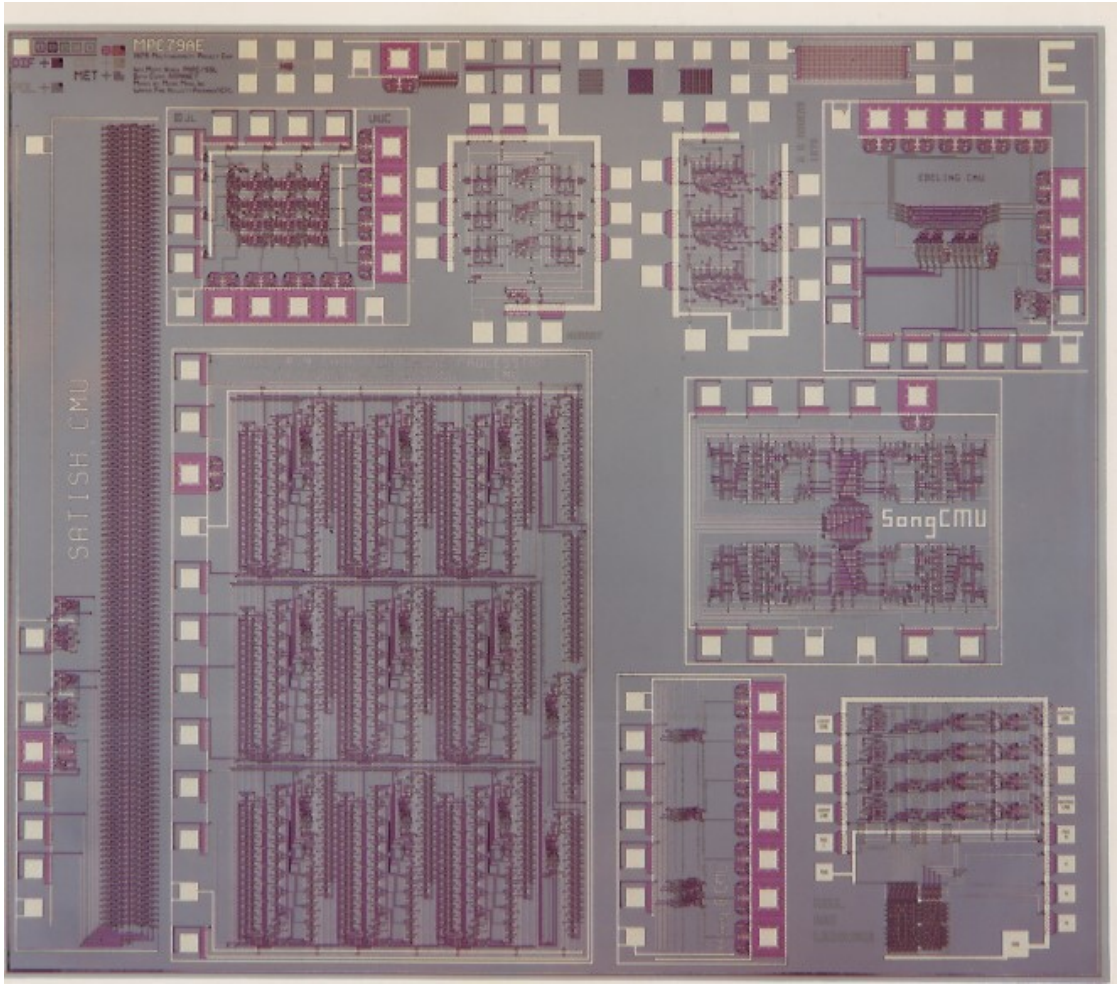


Source: Wikipedia



Source: Siang Wun Song

Pastilha multiprojeto - fabricada pela Xerox PARC



Pastilhas projetadas no curso de VLSI dado por Bob Sproull na CMU. Especificação em CIF enviada à Xerox por email para fabricação. A pastilha fabricada enviada de volta por correio (não por email :-(

Método formal para sintetizar arrays sistólicos


Patrice Quinton


Yves Robert


Dan Moldovan


José Fortes

- A novidade de projetar arrays sistólicos despertou enorme interesse na comunidade, dando origem a proposta de arrays sistólicos para os mais variados problemas.
- Em 1989, Quinton e Robert mostraram que, através de uma especificação em recorrências uniformes (laços aninhados) e transformação de dependências, pode-se sintetizar um array sistólico correspondente.

Quinton, P. & Robert, Y. Algorithmes et architectures systoliques*, Masson, Paris, 1989.*

- Um método semelhante foi descoberto por Dan Moldovan e José Fortes. Essas importantes descobertas formalizaram a obtenção de arrays sistólicos.

Array sistólico 1978 - a moda vai e volta

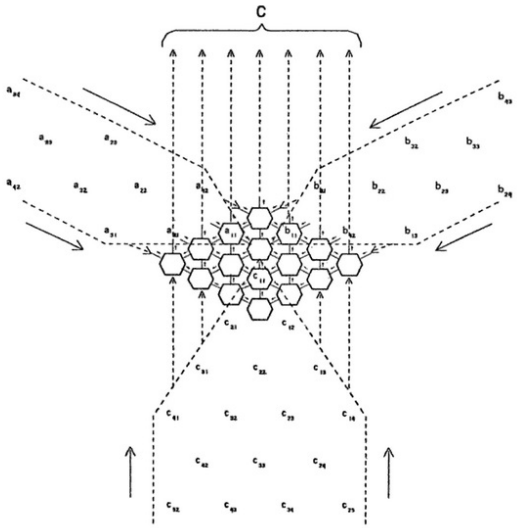
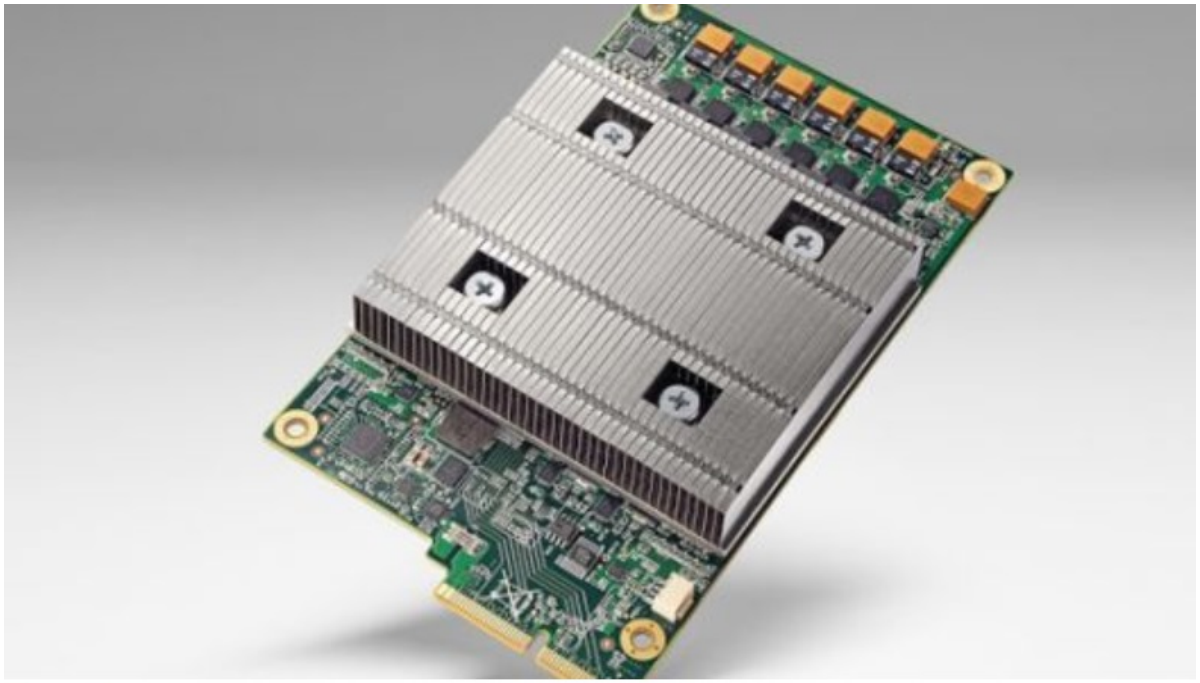


Figure 4-2: The hex-connected systolic array for the matrix multiplication problem in Figure 4-1.

- Proposto em 1978, array sistólico despertou enorme interesse na época.
- Mas com o tempo a moda passou e ficou latente durante quase trinta anos.
- Até que ressurge em 2016 pela Google TPU (Tensor Processing Unit).

A moda vai e volta...

Google TPU - Tensor Processing Unit 2016 - 2018

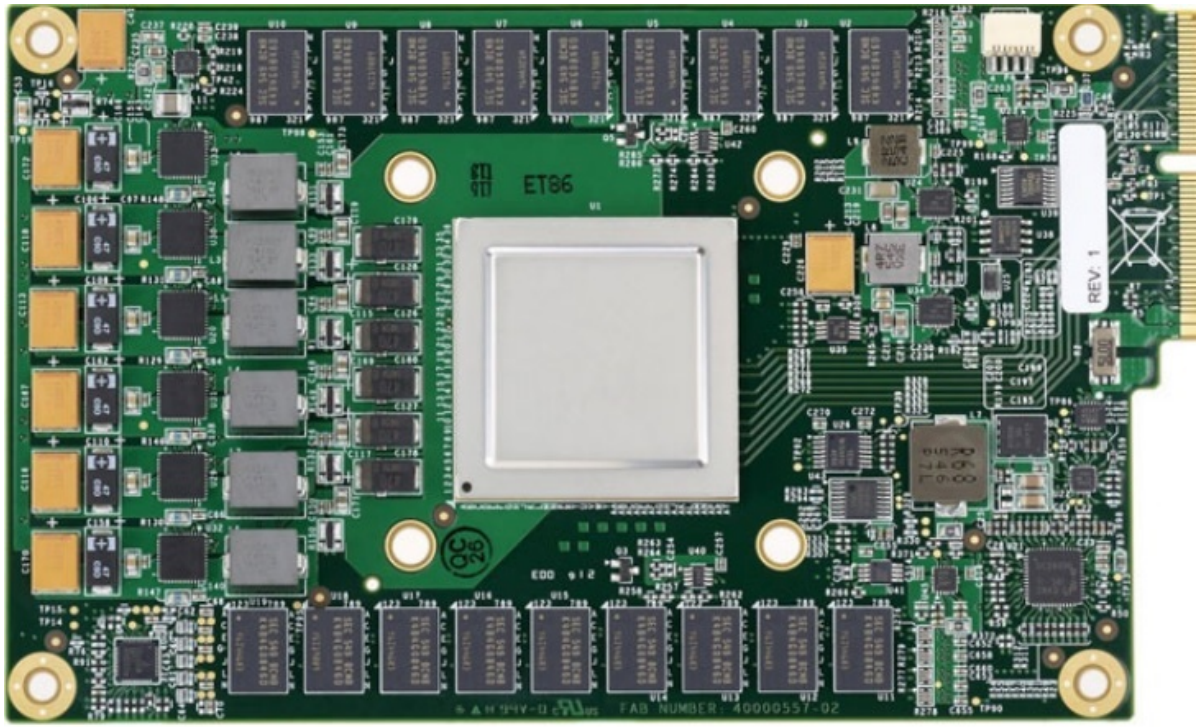


Source: Google

Google's tensor processing unit or TPU.

- Array sistólico ressurge na figura da Google TPU (Tensor Processing Unit) que é um array sistólico usado em Google Search, Google Street View, Google translate para acelerar as computações de redes neurais em aprendizado de máquina.

Google TPU - Tensor Processing Unit 2016



Google's first TPU

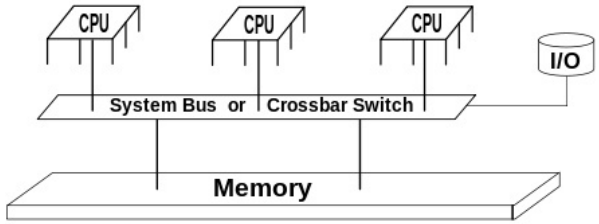
- Primeira geração TPU (2016): um 256 $\times$ 256 array sistólico que realiza multiplicação de matrizes de números inteiros de 8 bits, e operação de convolução.

Google TPU - Tensor Processing Unit 2018

- Segunda geração TPU (maio 2017): multiplicação de matrizes em ponto flutuante, com desempenho de 11,5 PetaFLOPS, usada no treinamento e inferência em redes neurais para aprendizado de máquina.
  em modelos de aprendizado de máquina.
- Terceira geração TPU (maio 2018): oito vezes mais rápido que TPU da segunda geração.

An in-depth look at Google's first Tensor Processing Unit (TPU). Kaz Sato (Staff Developer Advocate, Google Cloud), Cliff Young (Software Engineer, Google Brain), David Patterson (Distinguished Engineer, Google Brain) May 12, 2017.

Modelo PRAM - Parallel Random Access Machine



Source: Wikipedia

- Memória compartilhada serve para comunicação entre processadores. Adota modelo SIMD.
- Para um problema de tamanho n, o número de processadores pode depender de n.
- Acesso à memória leva uma unidade de tempo.
  - EREW (Exclusive Read Exclusive Write): uma mesma posição de memória só pode ser lida ou escrita por um processador.
  - CREW (Concurrent Read Exclusive Write): vários processadores podem ler uma mesma posição de memória, mas não podem escrever na mesma posição de memória.
  - CRCW (Concurrent Read Concurrent Write): leitura e escrita concorrente possíveis. Uma regra para escrita concorrente é só pode haver escrita concorrente quando todos escrevem o mesmo valor.

Modelo PRAM - Parallel Random Access Machine

Exemplo: Obter o mínimo de n números.

Algoritmo sequencial: $O(n)$ comparações são necessárias.

E no PRAM?

Resposta:Tempo constante.

Sim, isso mesmo. Para qualquer n, digamos $n = 10^{99}$.

Modelo PRAM - Parallel Random Access Machine

Exemplo: Obter o mínimo de n números.

Algoritmo sequencial: $O(n)$ comparações são necessárias.

E no PRAM?

Resposta:Tempo constante.

Sim, isso mesmo. Para qualquer n, digamos $n = 10^{99}$.

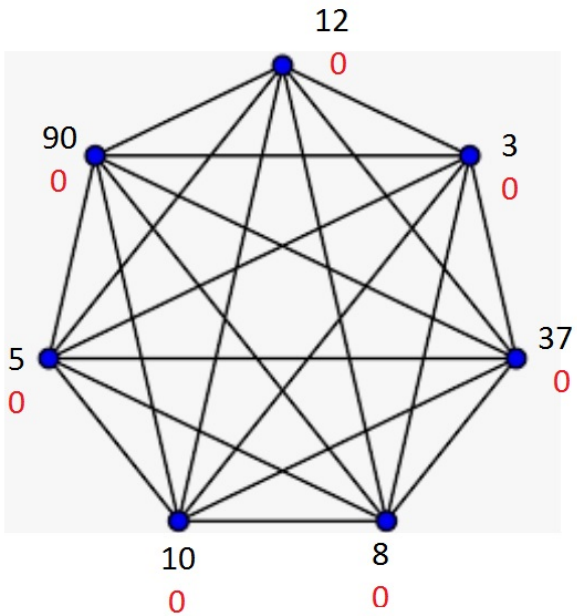
Modelo PRAM - Parallel Random Access Machine

O mínimo de n números pode ser obtido em tempo constante. no modelo PRAM-CRCW usando $O(n^2)$ processadores.

Seja $n = 7$. Usamos $n(n-1)/2 = 21$ processadores. Os dados estão no vetor *Dado*. O vetor *Resultado* contém 0 no início.

Dado	12	3	37	8	10	5	90
Resultado	0	0	0	0	0	0	0

Cada um dos 21 processadores compara um par de números *Dados*[i] e *Dados*[j].



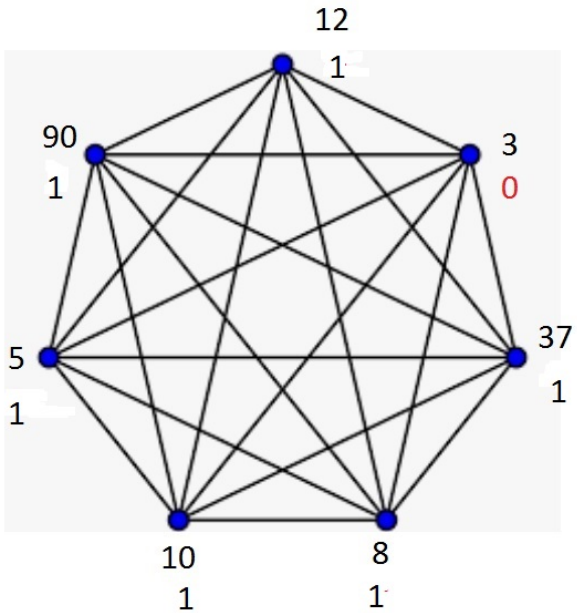
Modelo PRAM - Parallel Random Access Machine

O mínimo de n números pode ser obtido em tempo constante. no modelo PRAM-CRCW usando $O(n^2)$ processadores.

Seja $n = 7$. Usamos $n(n-1)/2 = 21$ processadores. Os dados estão no vetor *Dado*. O vetor *Resultado* contém 0 no início.

Dado	12	3	37	8	10	5	90
Resultado	1	0	1	1	1	1	1

Cada processador: Se *Dados*[i] > *Dados*[j] então faz *Resultado*[i] $= 1$. O mínimo é *Dado*[k] cujo *Resultado*[k]=0.



Renascimento de PRAM para uso em GPUs

- Muitos PRAM algoritmos têm complexidade poli-logaritmica: $T(n) = O((\log n)^k)$ para alguma constante k.
- O conjunto de problems de decisão decidíveis em tempo poli-logarítmico formam a classe NC (*Nick's Class* - classe de Nick Pippenger).
- Embora não realístico na prática, o modelo PRAM é importante para a teoria da complexidade.
- Mais recentemente, com o uso cada vez mais popular das GPUs (Graphical Processing Units), que adota o modelo de execução SIMD, vários trabalhos propõe adaptar algoritmos PRAM para GPUs.

Jürgen Brenner, Jörg Keller, and Christoph Kessler. Executing PRAM Programs on GPUs. Procedia Computer Science, Volume 9, 2012, pp. 1799-1806.

Frank Dehne and Kumanan Yogaratnam. Exploring the Limits of GPUs With Parallel Graph Algorithms. Cornell Univesity Library, arXiv:1002.4482, 2010.

Modelo BSP - Bulk Synchronous Parallel

Leslie Valiant

Source: Wikipedia

- BSP (Bulk Synchronous Parallel) - Leslie Valiant (Turing Award 2010): um modelo realístico levando em consideração comunicação e sincronização.
- Algoritmo expresso em superpassos (fase de computação e fase de comunicação) separados por barreiras de sincronização.
- Parâmetros usados: número de processadores, número de mensagens no superpasso, largura da banda, latência.

Modelo CGM - Coarse-Grained Multicomputer

Frank Dehne

Source: Carleton University

- CGM (Coarse-Grained Multicomputer) - Frank Dehne.
- Como BSP: algoritmo expresso em superpassos (rodada de computação e rodada de comunicação) separados por barreiras de sincronização.
- Parâmetros usados: tamanho da entrada n e número de processadores p.
- Objetivo: Minimizar o número de superpassos e o tempo total.
- SPMD (*Same Program Multiple Data*).

Modelo CGM - Coarse-Grained Multicomputer

Frank Dehne

Source: Carleton University

Algorithmica (2002) 33: 183–200
DOI: 10.1007/s00453-001-0109-4

Algorithmica

Efficient Parallel Graph Algorithms for Coarse-Grained Multicomputers and BSP[1]

F. Dehne,[2] A. Ferreira,[3] E. Cáceres,[4] S. W. Song,[5] and A. Roncato[6]

Abstract. In this paper we present *deterministic* parallel algorithms for the *coarse-grained multicomputer* (CGM) and *bulk synchronous parallel* (BSP) models for solving the following well-known graph problems: (1) list ranking, (2) Euler tour construction in a tree, (3) computing the connected components and spanning forest, (4) lowest common ancestor preprocessing, (5) tree contraction and expression tree evaluation, (6) computing an ear decomposition or open ear decomposition, and (7) 2-edge connectivity and biconnectivity (testing and component computation). The algorithms require $O(\log p)$ communication rounds with linear sequential work per round (p = no. processors, N = total input size). Each processor creates, during the entire algorithm, messages of total size $O(\log(p)(N/p))$.

The algorithms assume that the local memory per processor (i.e., N/p) is larger than p^{ε}, for some fixed $\varepsilon > 0$. Our results imply BSP algorithms with $O(\log p)$ supersteps, $O(g \log(p)(N/p))$ communication time, and $O(\log(p)(N/p))$ local computation time.

It is important to observe that the number of communication rounds/supersteps obtained in this paper is independent of the problem size, and grows only logarithmically with respect to p. With growing problem size, only the sizes of the messages grow but the total number of messages remains unchanged. Due to the considerable protocol overhead associated with each message transmission, this is an important property. The result for Problem (1) is a considerable improvement over those previously reported. The algorithms for Problems (2)–(7) are the first practically relevant parallel algorithms for these standard graph problems.

Key Words. Coarse grained parallel computing, Graph algorithms.

Problema	Número superpassos
List ranking	$O(\log p)$
Euler tour	$O(\log p)$
Componentes conexos	$O(\log p)$
Lowerest common ancestor	$O(\log p)$
Contração árvore	$O(\log p)$
Open ear decomposition	$O(\log p)$
bi-conectividade	$O(\log p)$

Modelo CGM - Coarse-Grained Multicomputer

- Usando o modelo CGM, desenvolvemos algoritmos paralelos escaláveis para várias classes de problemas:
  - Problemas em grafos: fecho transitivo, componentes conexos, árvore geradora mínima, teste de planaridade, etc.
  - Processamento de cadeias de caracteres, com aplicações em Bioinformática: similaridade de cadeias, subsequência comum mais longa, comparação de dois genomas, subsequência máxima, etc.

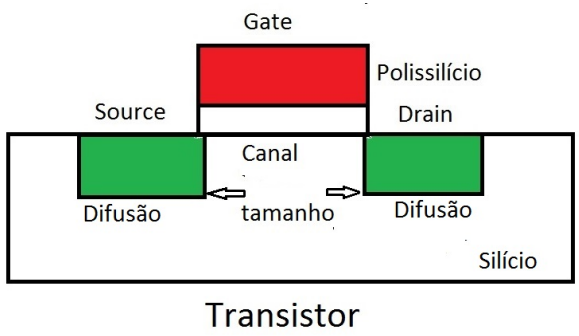
TERCEIRA LIÇÃO: Para publicar um trabalho não basta ter uma boa ideia. Uma boa ideia é como um diamante. Precisa ser cuidadosamente lapidado e polido para ser apreciado e mostrar toda a sua beleza.

Modelo CGM - Coarse-Grained Multicomputer

- Usando o modelo CGM, desenvolvemos algoritmos paralelos escaláveis para várias classes de problemas:
  - Problemas em grafos: fecho transitivo, componentes conexos, árvore geradora mínima, teste de planaridade, etc.
  - Processamento de cadeias de caracteres, com aplicações em Bioinformática: similaridade de cadeias, subsequência comum mais longa, comparação de dois genomas, subsequência máxima, etc.

TERCEIRA LIÇÃO: Para publicar um trabalho não basta ter uma boa ideia. Uma boa ideia é como um diamante. Precisa ser cuidadosamente lapidado e polido para ser apreciado e mostrar toda a sua beleza.

Comentários finais



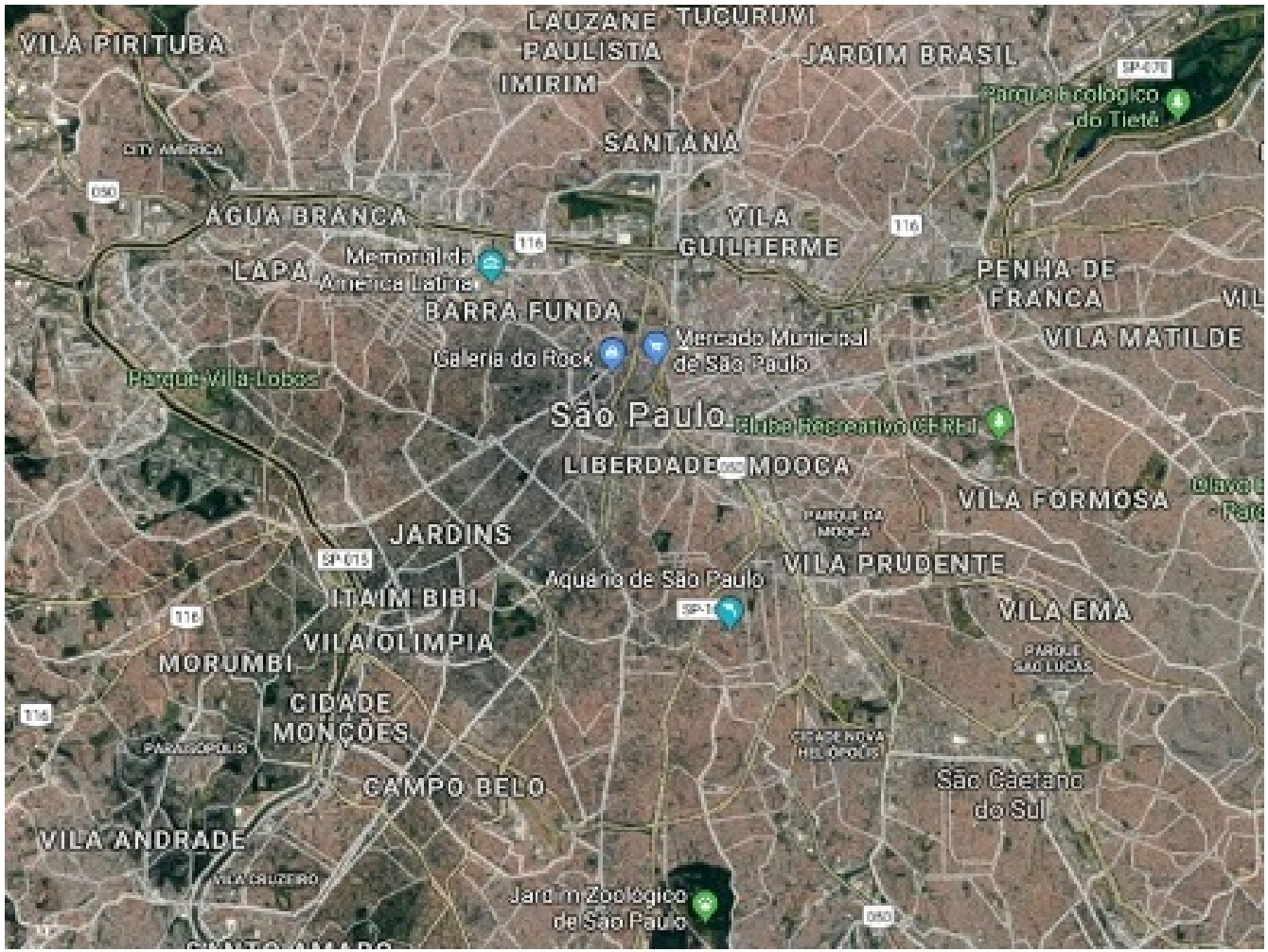
Source: Siang Wun Song

- A evolução da Computação Paralela tem a ver com o avanço da tecnologia VLSI.
- O tamanho característico (*feature size*) de um transistor num circuito integrado passou de 5 micrômetros em 1978 para 7 nanômetros em 2018.
- Se uma dimensão (1D) na pastilha diminui 5000/7 = 714 vezes, então temos uma redução de $714^2 = 510.000$ na área (2D) da pastilha.

Ano	Tamanho transistor	Redução (1D)	Redução na área (2D)
1978	5 micrômetros		
2018	7 nanômetros	714 vezes	510.000 vezes

Comentários finais

Suponha que a pastilha de 1978 continha, ao invés de circuitos VLSI, uma região geográfica de 1.000 km^2.



Source: Google Maps

Comentários finais

A mesma pastilha hoje pode conter toda a terra, com uma área de 510.000.000 km^2. Isso ilustra o avanço da VLSI.

Source: Google Maps

Comentários finais

Esse avanço na tecnologia VLSI explica por que hoje Google pode fabricar a TPU.

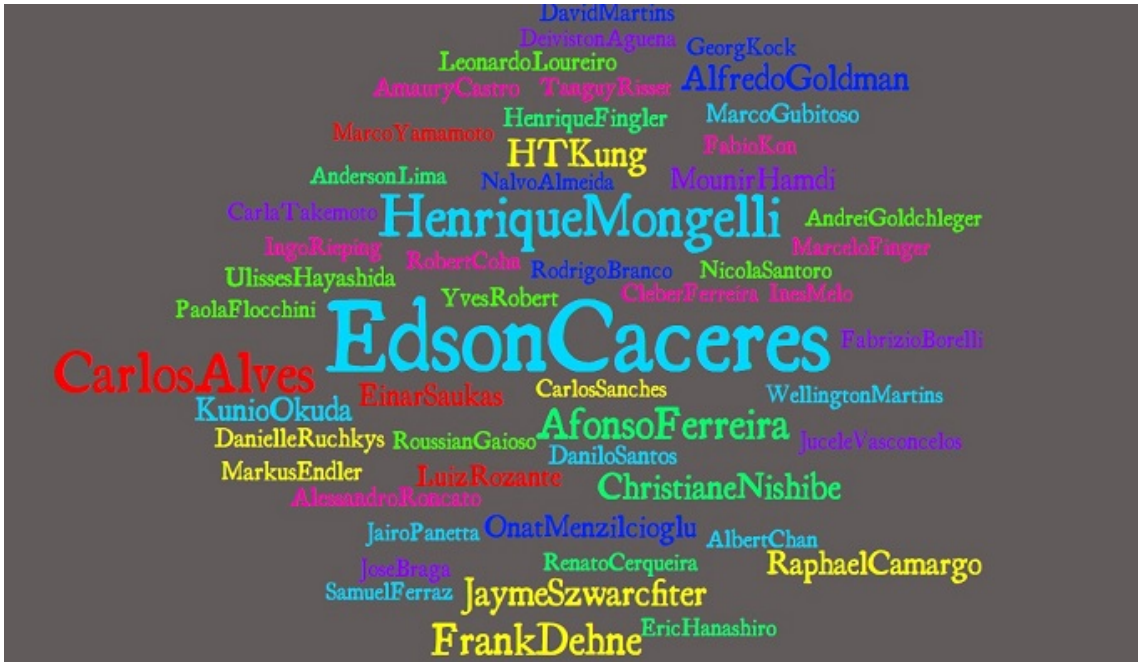
Source: Google Maps

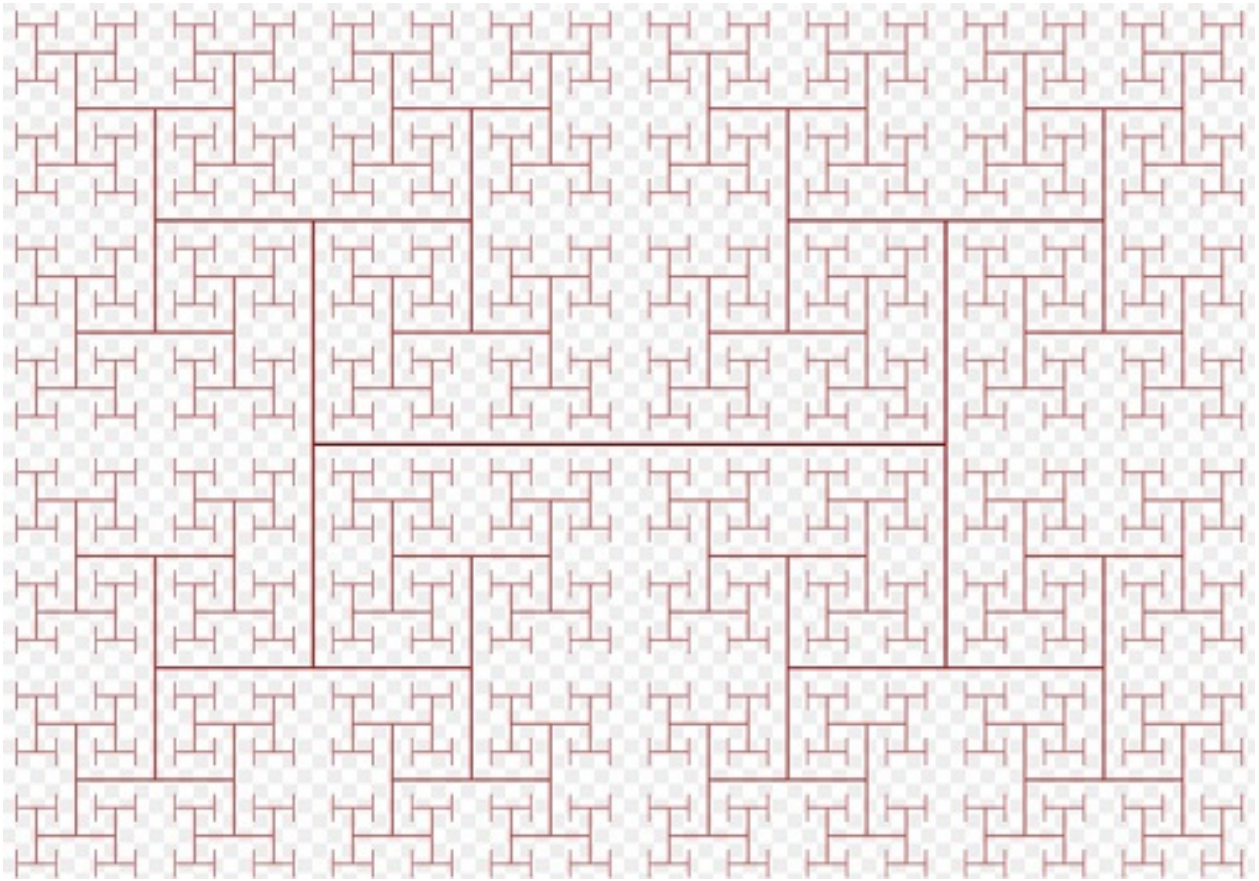
Comentários finais

awrence Livermore National Laboratory

- Hoje o computador mais veloz do mundo é Summit - E.U.A. com total de 3.120.000 *cores* ou processadores.
- Tem uma velocidade de pico 187,6 PetaFLOPS.
- Provavelmente em menos de 5 anos espera-se chegar a *Exascale Computing* (1 ExaFLOPS = 1024 PetaFLOPS), capaz de realizar um quintilhão ou 10^{18} operações aritmética por segundo.
- O que poderemos fazer com todo esse potencial computacional? As oportunidades serão imensas.

Meus colaboradores





Obrigado!

Quarenta+ anos de Computação Paralela - uma visão pessoal

Siang Wun Song

<song@ime.usp.br>

2.o Encontro Paulista de Pós-Graduandos em Computação
Campinas, 8 de novembro de 2018

Slides em https://www.ime.usp.br/∼song