Workshop do Núcleo de Apoio à Pesquisa em Software Livre
São Carlos, 16 a 17 de Outubro de 2014.

# EXECUTANDO APLICAÇÕES BSP NO OPENSTACK

Thiago Kenji Okada
IME-USP
E-mail: thiagoko@ime.usp.br

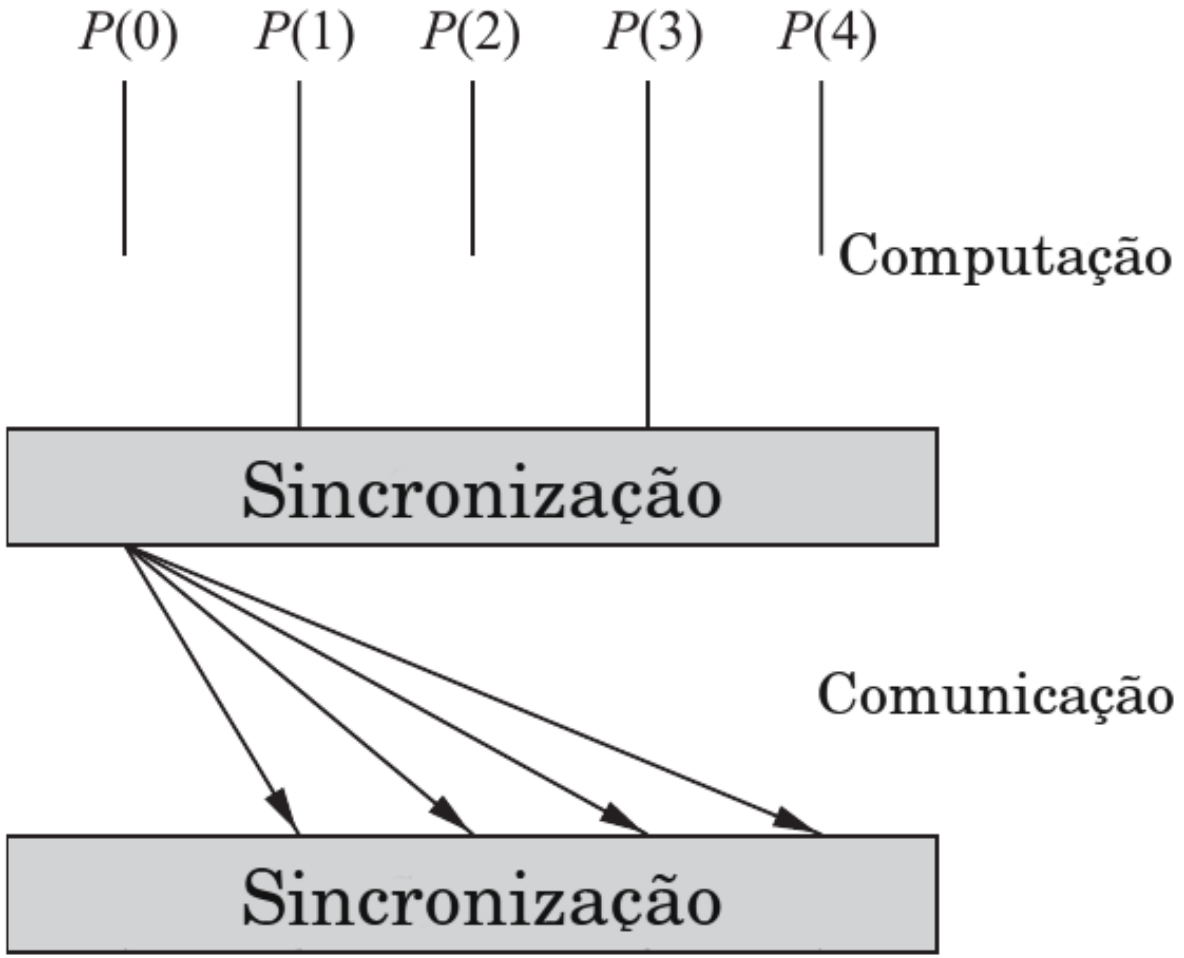
# Cronograma

- O modelo BSP
- Computação em nuvem e BSP
  - Escalonando aplicações BSP na nuvem
- Uso de código-aberto
- Por que o OpenStack?

# O modelo BSP

- Modelo de programação paralela proposto por Leslie Valiant (1989);
- Composto de um modelo computacional e um modelo de algoritmo.
  - Um **computador BSP** é composto de:
    - Uma conjunto de processadores, cada um com a sua memória;
    - Uma rede de comunicação, para troca de mensagens;
    - Um mecanismo de sincronização entre processadores.
  - Um **algoritmo BSP** é composto de:
    - Uma sequência de super-passos, que podem ser tanto de comunicação como de computação.

# O modelo BSP


P(0)
P(1)
P(2)
P(3)
P(4)
Computação
Sincronização
Comunicação
Sincronização

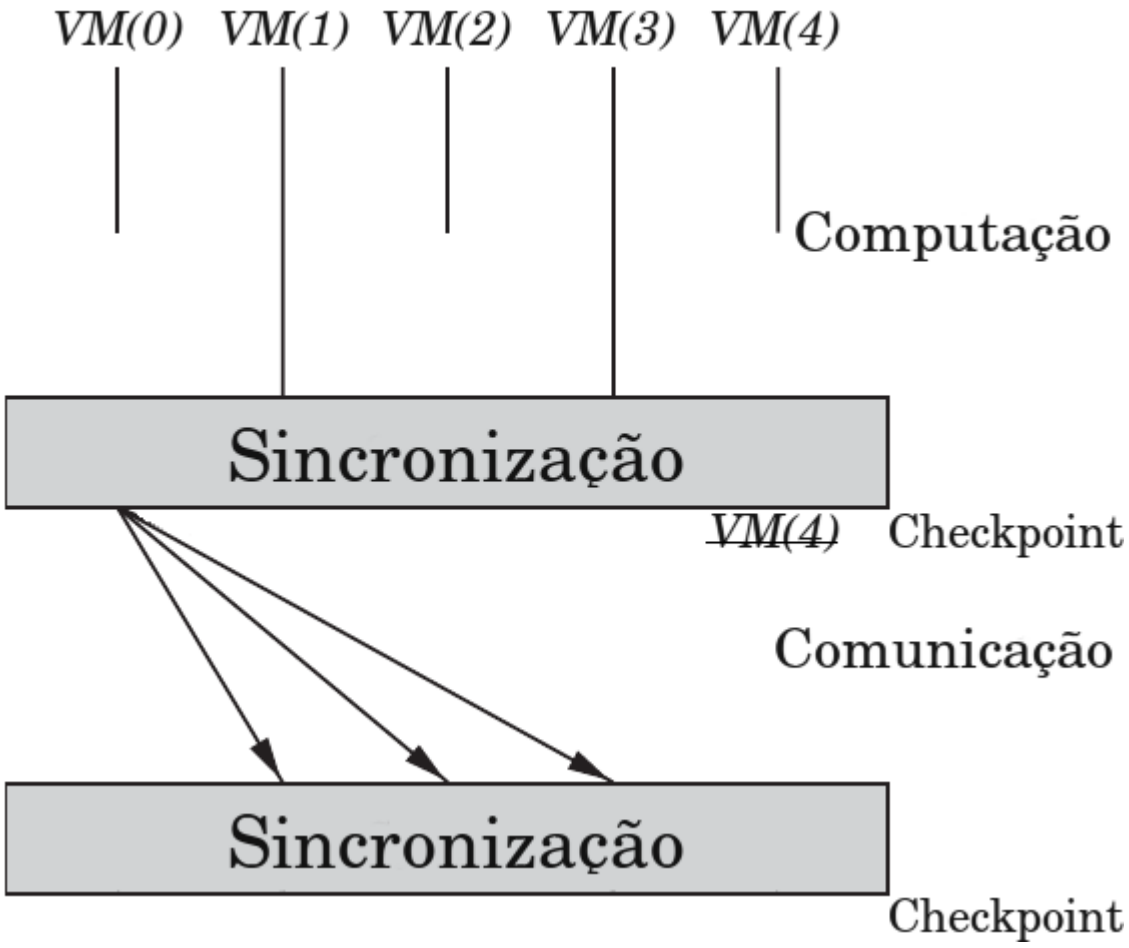
# Computação em nuvem e BSP

- O modelo BSP é tradicionalmente usado em máquinas locais ou *grids* computacionais;
- A ideia é usar computação em nuvem no lugar;
  - Problema: o ambiente de computação em nuvem é heterogêneo.
    - Porém o modelo BSP abstrai esses problemas;
    - E iremos usar um ambiente controlado para testes (usando uma nuvem OpenStack “privada”).
- Além disso, queremos escalonar essas aplicações BSP automaticamente.

# Escalonando aplicações BSP na nuvem

- Podemos alterar o número de recursos numa MV de forma (quase) dinâmica;
- Como fazer isso durante a execução de uma aplicação BSP?
  - Uso de *checkpoints*!
    - Ao fim de cada super-passo, armazenar os dados da aplicação, possivelmente (re-)escalonar os processos e continuar com o próximo super-passo.

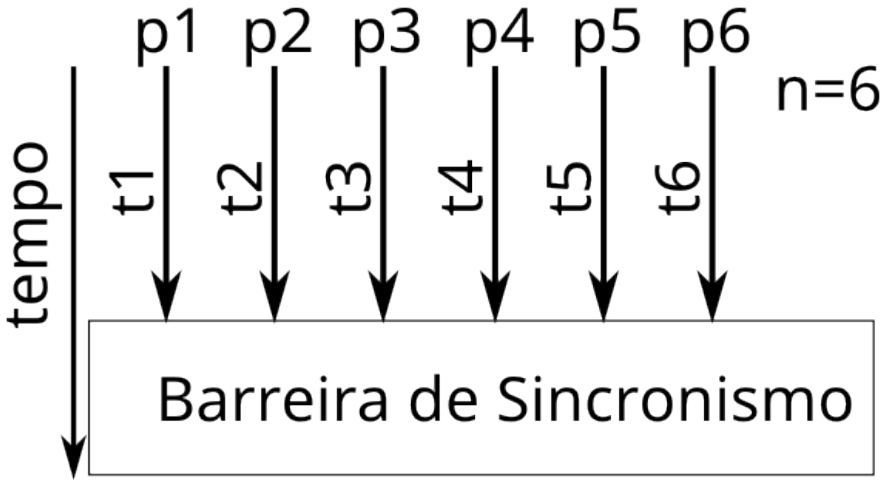
# O nosso modelo BSP

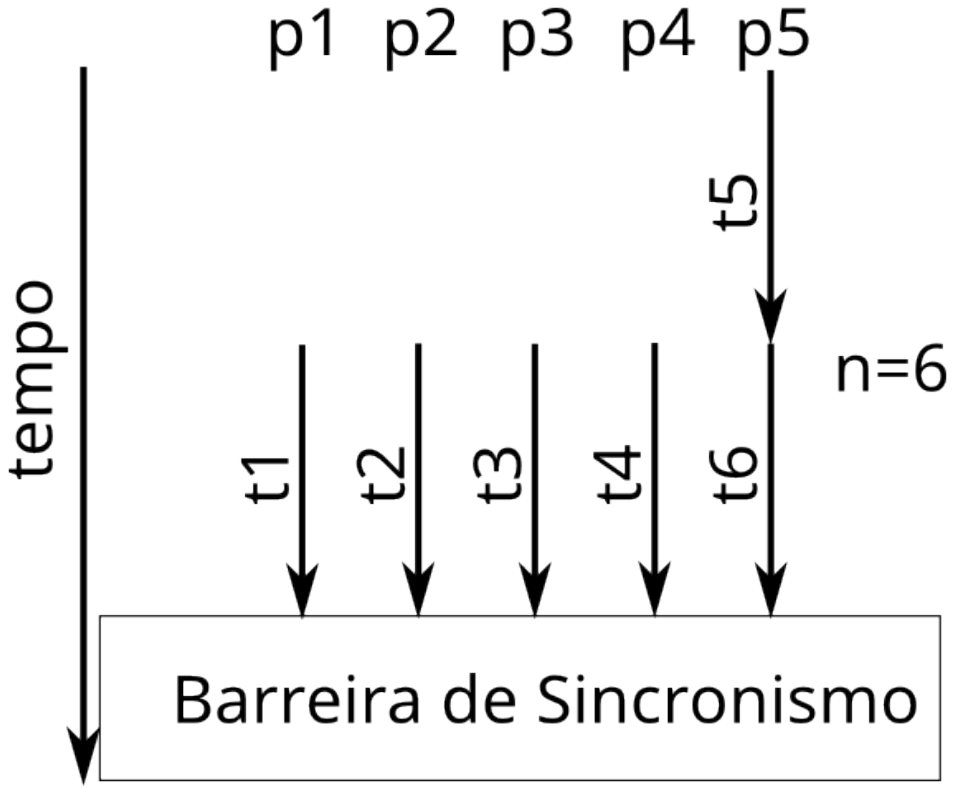


# Modelando aplicações BSP

- Uma super-passo no modelo BSP é executado em *t* tempo com *p* processadores;
- Se usarmos *p-1* processadores, o tempo dobra! E se usarmos p+1 processadores, o tempo continua o mesmo. Por que?

# Modelando aplicações BSP



# Modelando aplicações BSP

- Portanto, se quisermos reduzir o número de processadores, devemos usar *p/2^q*, onde *q* é o fator de *speedown* que queremos alcançar.
  - Isso é uma das coisas que queremos que o escalonador trate sozinho!
- Podemos pensar algo semelhante para aumentar a velocidade do programa, usando *2*q*p*.
  - Porém nem todos os programas escalam de maneira tão simples, por isso teremos que testar!

# Uso de código-aberto

- Uso do OpenStack como plataforma de computação em nuvem;
  - Além de ser de código-aberto (licença Apache), é escrito em Python ;).
- Para a escrita de aplicações BSP, será usado o OpenMPI;
  - Disponível para diversas linguagens, arquiteturas e sistemas operacionais.
- Linux, QEMU (KVM)...

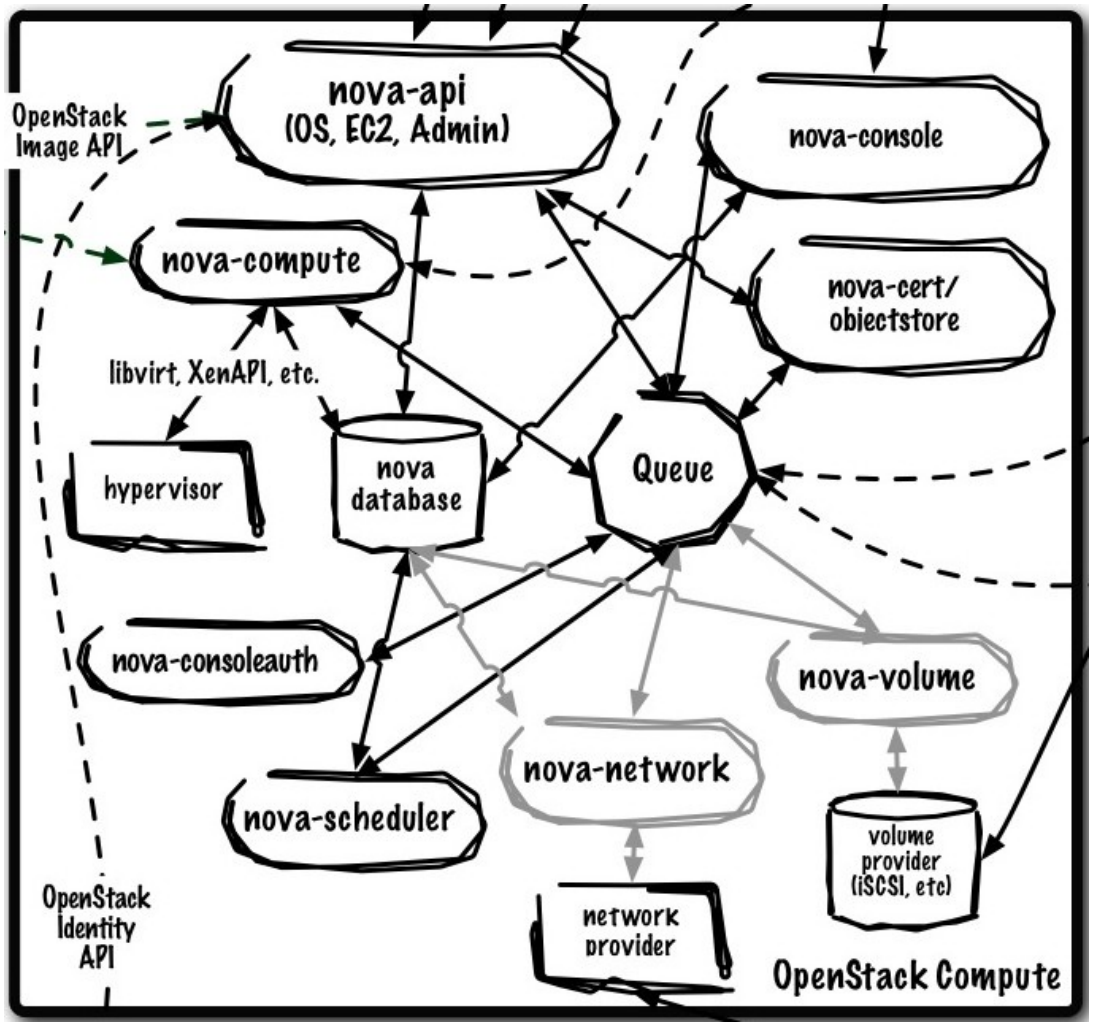
# Por que o OpenStack?

- Mais de 850 organizações participam do projeto, além de 9500 indivíduos independentes;
- Permite *self-hosting*;
- Código-aberto;
- Arquitetura modular.

# OpenStack Nova

- Provê a funcionalidade de computação na nuvem (IaaS);
- Alguns módulos importantes para esse trabalho:
  - *nova-api:* permite criação de MVs com as respectivas alocações de recursos;
  - *nova-scheduler:* determina em que máquinas físicas as Mvs serão instanciadas;
  - *nova-compute:* inicia e termina instâncias de Mvs.

# OpenStack Nova



# Fim da apresentação