Plánování úloh v dynamických, distribuovaných a heterogenních systémech

Dalibor Klusáček
xklusac(at)fi.muni.cz

PV177 Laboratoř sítí, 3. 3. 2010

Obsah

- O plánování
- Popis problému
- Klasické plánovací techniky
- Pokročilé plánovací techniky
- Návrh plánovacích algoritmů
  - Simulace plánování
  - Experimenty a datové sady

O plánování

- Intuitivní definice
  - Plánujeme nějaké „aktivity“
  - Cílem je „úspěšné dokončení“ (realizace) aktivit
  - Nejlépe tak, abychom byli „efektivní“
  - A přitom „flexibilní“
- Slova v uvozovkách si blíže nadefinujeme
- Budeme se zabývat plánováním úloh na stroje

Popis problému

- Úlohy
- Zdroje
- Kritéria (optimalizační)
- Vlivy prostředí
- Metody řešení

Úlohy

- Úloha je uživatelská aplikace
- Úloha má vlastnosti
  - Doba trvání
  - Doba příchodu (kdy ji uživatel pošle)
- Úloha má požadavky
  - Počet procesorů
  - Softwarové knihovny
  - Velikost paměti
  - Síť
  - …

Zdroje

- V našem případě jsou zdroje zejména:
  - Stroje a procesory
- A pak také:
  - Disková kapacita
  - Velikost paměti
  - Počítačová síť
  - Dostupné softwarové licence
  - …
- Zdroje se prováděním úloh „spotřebovávají“

Kritéria

- Optimalizační kritéria slouží k:
  - Definici toho co chceme dosáhnout
  - Vyhodnocení kvality řešení (hodnota tzv. objektivní fce.)
  - Řízení běhu plánovacích algoritmů
  - Srovnání různých plánovacích algoritmů
- Např:
  - Počet zpožděných úloh, vytížení strojů, tzv. slowdown, doba čekání na spuštění, celkový čas provádění, atd.

Statické vs. dynamické plánování

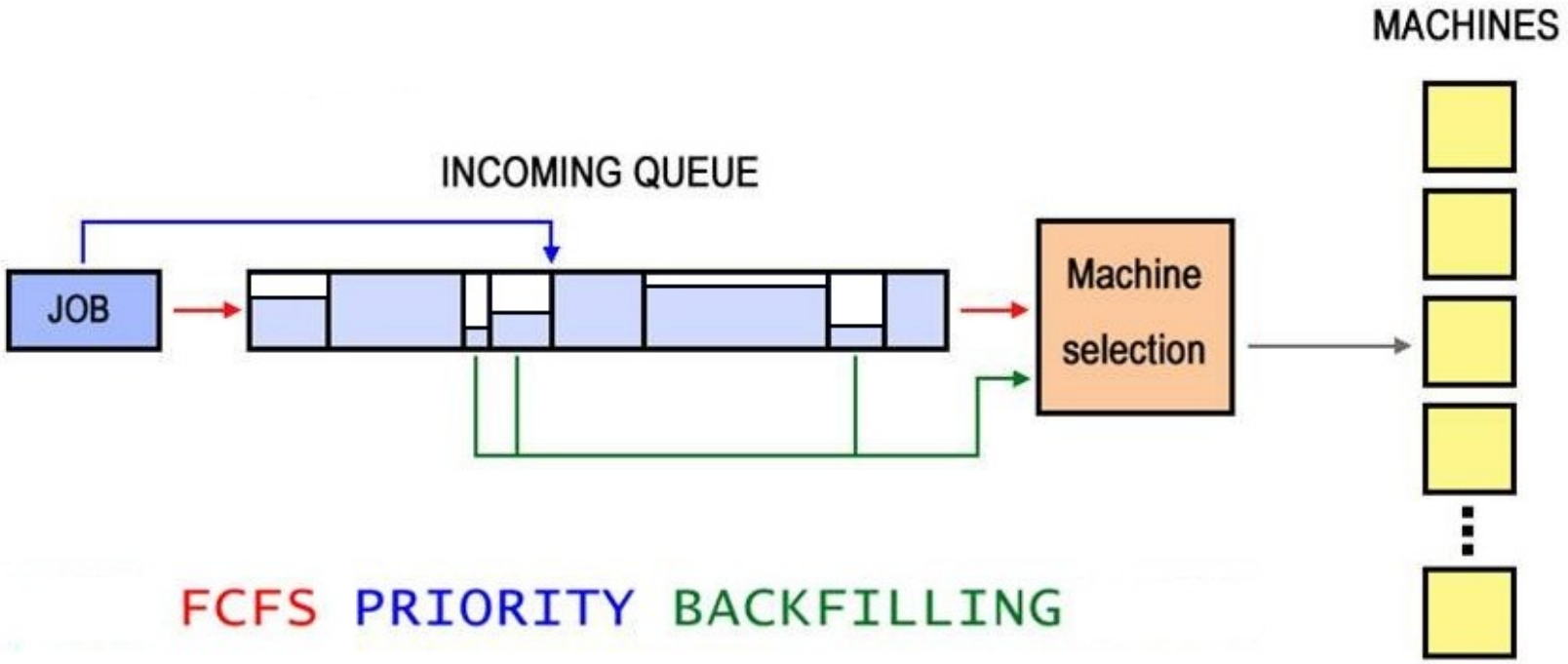
- Statické plánování:
  - Známe stroje
  - Známe úlohy
  - Naráz je naplánujeme
- Dynamické
  - Počty úloh i strojů se v čase mění
  - Plánování probíhá paralelně se „změnami v systému“
  - Důležitá je efektivita a rychlost

Metody řešení

- Pro statické problémy lze teoreticky využít mnoho různých technik
  - Integer programming
  - Heuristiky
  - Fronty
  - řídící pravidla
  - CSP, úplné řešiče
  - …
- V dynamickém případě je nejčastější
  - Plánování pomocí front(y)
  - Plánování pomocí tvorby rozvrhu

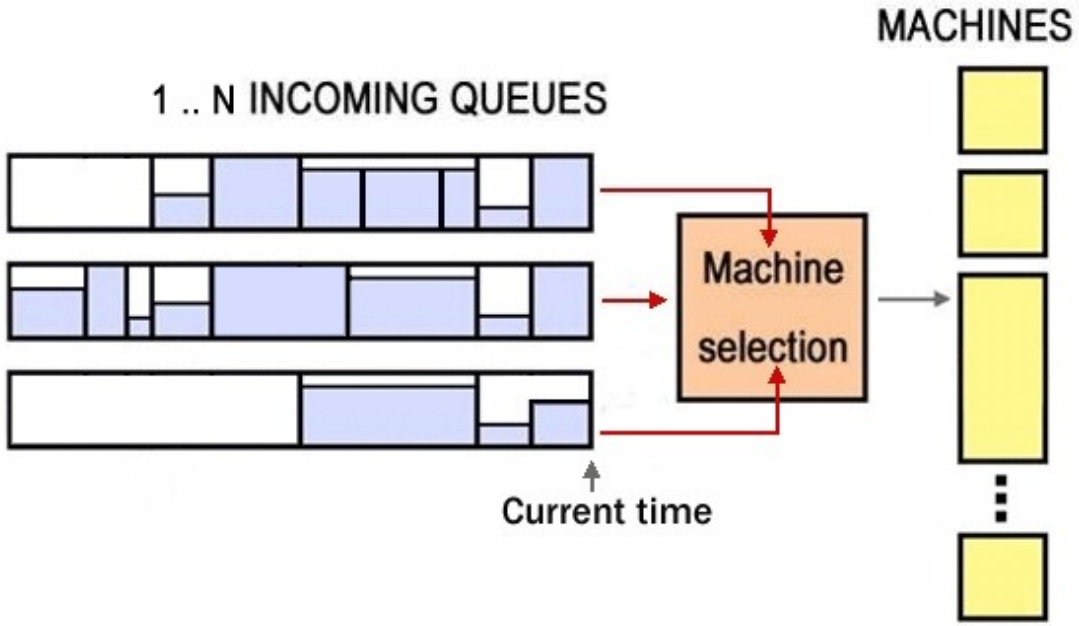
Plánování pomocí fronty úloh

- 1 fronta pro úlohy
- Úlohy přichází do fronty
- Je-li volný stroj, vybere se úloha z fronty a pošle na stroj
- Algoritmy se liší tím, jak zařazují a vybírají úlohy z fronty



Plánování pomocí front úloh

- N front pro úlohy
- Úlohy přichází do **některé** z front
- Je-li volný stroj, vybere se úloha z **některé** fronty a pošle se na stroj
- Algoritmy se liší tím:
  - Jak zařazují a vybírají úlohy z fronty
  - Jakým způsobem pracují s jednotlivými frontami

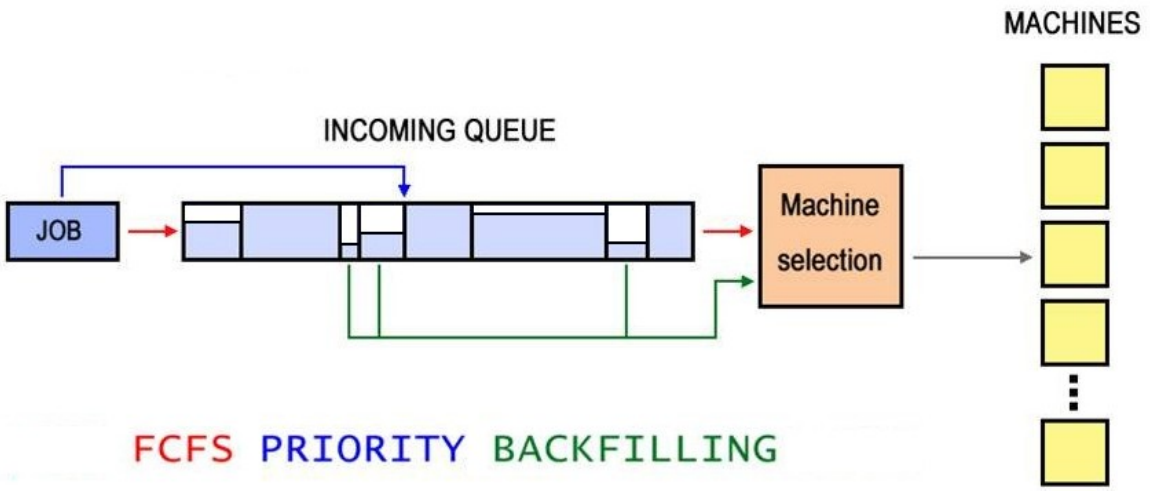


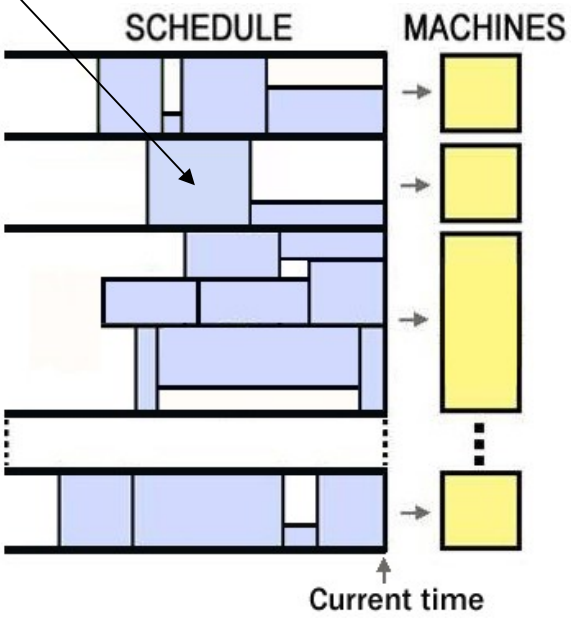
„Realistické“ plánování

- Výpadky strojů
  - Co dělat se zabitými úlohami?
  - Co dělat s úlohami, které najednou nemají kde běžet?
- Neznámá doba trvání úloh
  - Mnoho algoritmů vyžaduje alespoň odhad doby trvání
  - **Jak jej získat?**
- Omezení daná uživatelem
  - Na stroje, na soft. licence, na síť, ..
- Omezení daná administrátorem
  - Kdo sem může a kdo ne
  - Kdo má jakou prioritu

Plánování založené na rozvrhu úloh

- Nemáme žádné fronty
- Místo nich budujeme tzv. **rozvrh**
- Jde de facto o „plán budoucího výpočtu úloh“





Rozvrh – vlastnosti

- Každá úloha zná své „souřadnice“:
  - Na kterém stroji má běžet
  - Která CPU se budou alokovat
  - Kdy má (nejspíš) začít
  - Kdy má běžet
  - Kdy má (nejspíš) skončit

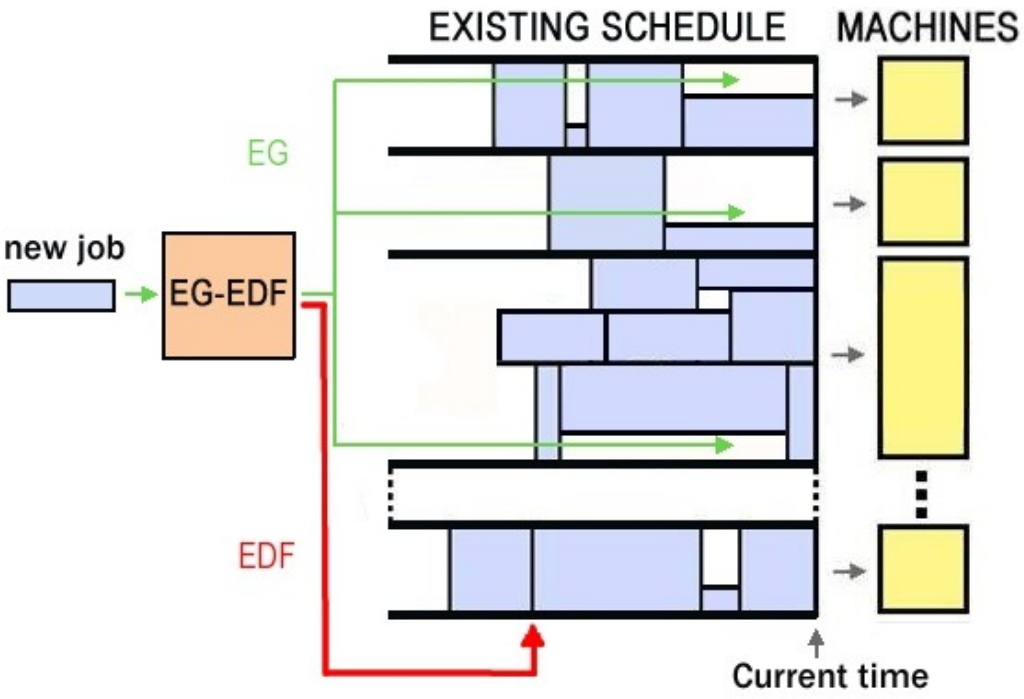
- Výhody vs. Nevýhody
- **Co myslíte?**

Principy výhod a nevýhod

- Nevýhody:
  - Je třeba znát co možná nejpřesněji dobu trvání úloh
  - Bez jakékoliv znalosti o době trvání nelze rozvrh korektně zkonstruovat
  - **Proč?**
- Výhody:
  - Existuje široká škála optimalizačních technik pro „vylepšování“ rozvrhů
  - Rozvrh samotný lze snadno „vyhodnotit“, frontu nikoliv

Policies (řídící pravidla)

- Jednoduché heuristiky pro tvorbu rozvrhu
  - Vezmi příchozí úlohu
  - Podívej se na aktuální rozvrh
  - Vytipuj si vhodná místa kam dát úlohu
  - Vyzkoušej je a vyber nejlepší
- Rychlé
  - srovnatelné s kvalitními frontovými algoritmy

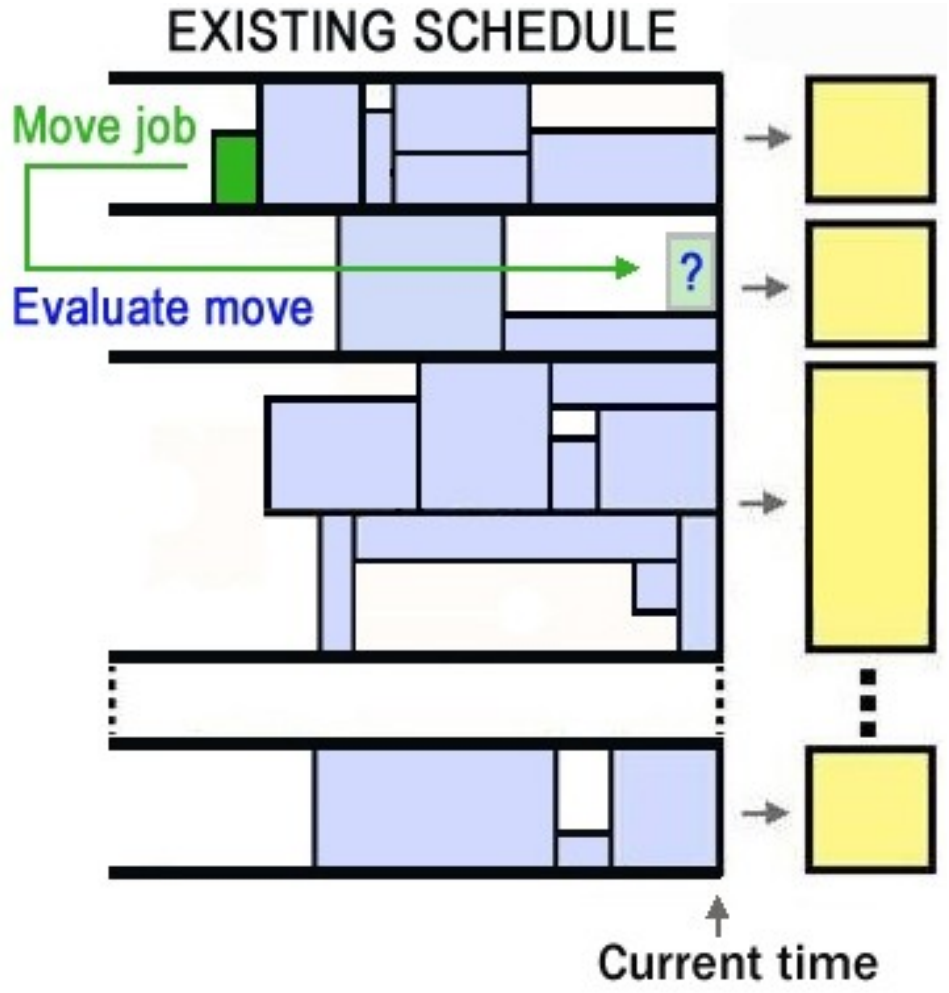


Optimalizace

- Optimalizace probíhá nad rozvrhem
- Provádíme úpravy rozvrhu (přesuny úloh)
- Zkoumáme jejich vliv na hodnotu objektivní funkce
- Cílem je zlepšení rozvrhu, který byl vygenerován „rychlým a hloupým“ postupem
- Optimalizujeme je-li čas
- Optimalizujeme je-li třeba
  - Např. narušení rozvrhu díky nepřesnému odhadu doby trvání

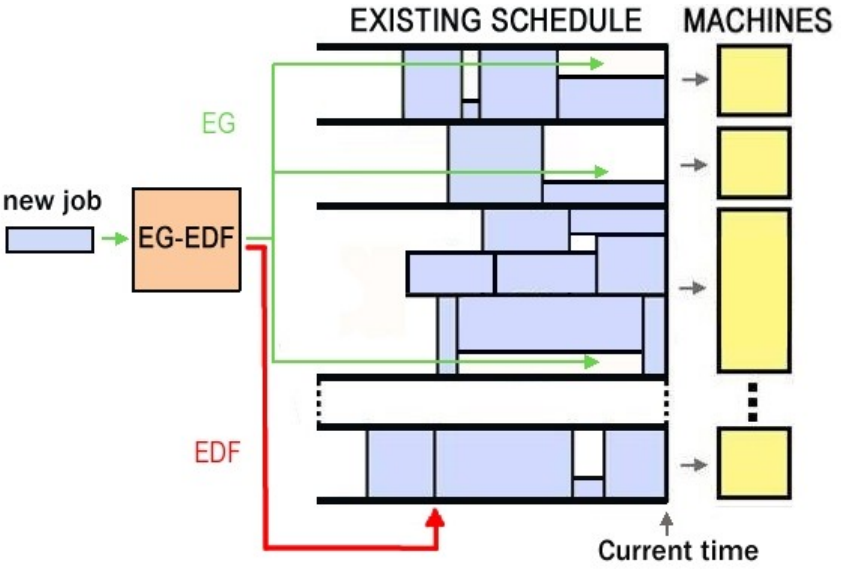
Lokální prohledávání

- Lokální změny
- Přesuny úloh
- Dostatečně rychlé
- V řádu stovek milisekund dokáže vylepšit relativně veliký rozvrh (stovky CPU a stovky úloh)



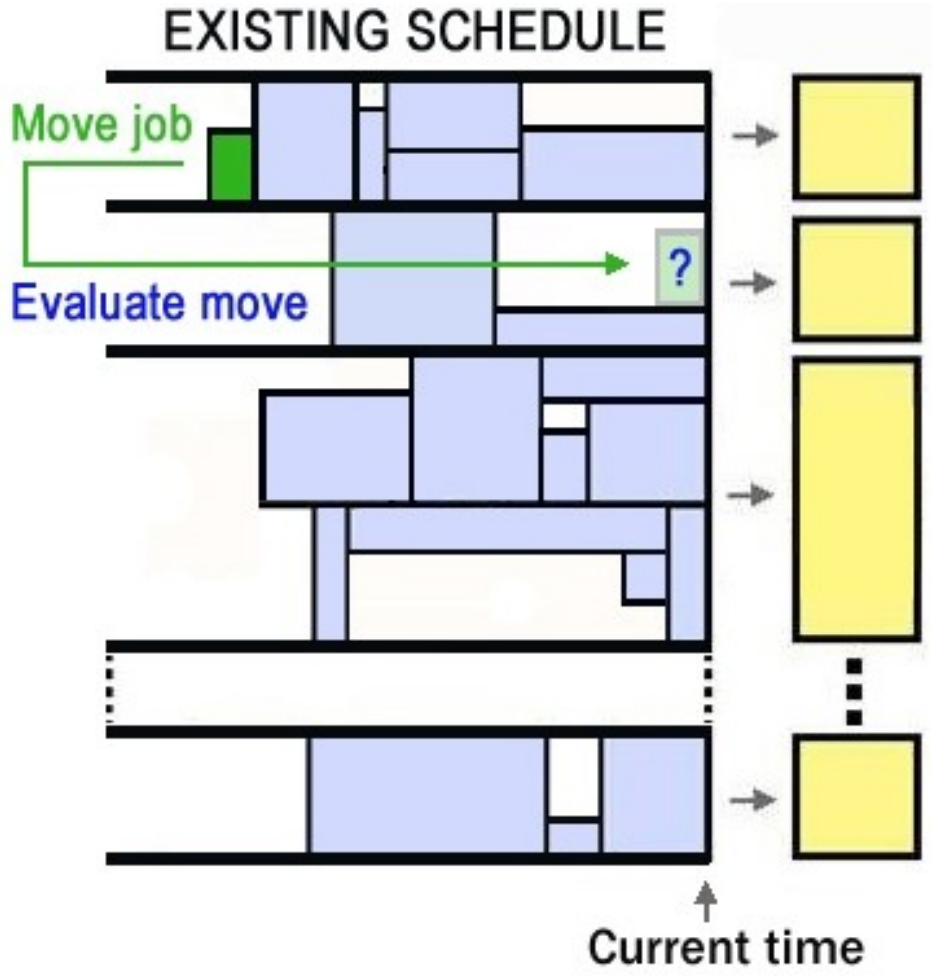
Efektivní plánování – inkrementální přístup

- Inkrementální přístup tvorby rozvrhu je klíčem k rychlosti těchto technik
- Příchod nové úlohy nesmí způsobit přepočítávání rozvrhu pro všechny úlohy (což se jinak běžně dělá)
- Místo toho využijeme existující řešení a novou úlohu do něj „nenásilně“ zařadíme



Neznámá doba trvání úloh

- Je to komplikace
- Lze použít odhady
- Platí, že úloha je kratší než odhad
- V opačném případě je totiž zabita
- Rozvrh lze zkonstruovat, vznikají ale „díry“
- Ty lze efektivně zaplňovat například pomocí lokálního prohledávání



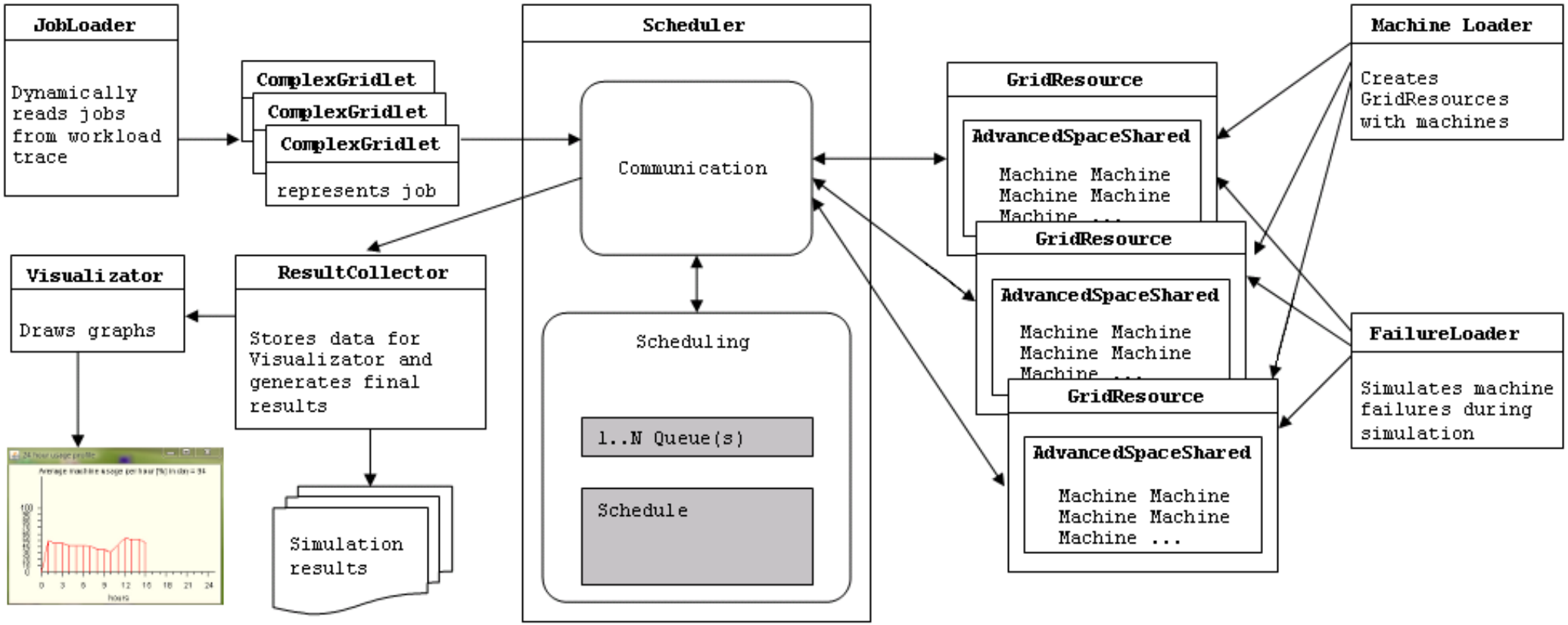
Simulace plánování

- Vývoj nových algoritmů je složitý
  - Je třeba testovat před nasazením
  - Opakovatelnost výsledků
  - Různá nastavení
  - Deterministický běh
  - V reálu nedosažitelné
  - **Víte proč?**
- Proto simulace plánování
  - Simulátor Gridového plánování vyvinutý na FI

Alea – Job Scheduling Simulator

- Napsaný v Javě nad balíkem GridSim
- Snadno rozšiřitelný
- Obsahuje implementace:
  - Obvyklých plánovacích algoritmů
  - Obvyklých objektivních funkcí
  - Parsery vstupních dat pro obvyklé formáty
  - Podporu export do obvyklých formátů (csv,png,jpg,…)
- Je optimalizován na vysokou rychlost simulace

Alea – Job Scheduling Simulator



Alea – Job Scheduling Simulator

Dostupná data

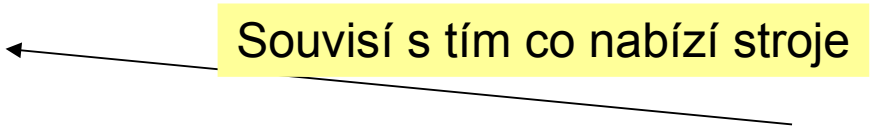
- Co a jak simulovat?
- Dostupná data:
  - Parallel Workloads Archive
  - Grid Workloads Archive
  - Failure Trace Archive
  - MetaCentrum workload trace
- Liší se mírou obsažených informací
  - Málo informací znamená nerealistické simulace
  - Bohaté sady umožní simulovat složitější a realističtější scénáře

Současný přístup

- Dnes je obvyklé simulovat:
  - Grid jako neměnný svět (N strojů s M procesory)
  - Stroje jako identické
    - Rozdíly pouze v rychlosti a počtu CPU
  - Úlohy jako identické
    - Rozdíly pouze v délce a požadovaném počtu CPU
- **Je to dostatečné?**
- **Jaká je realita?**

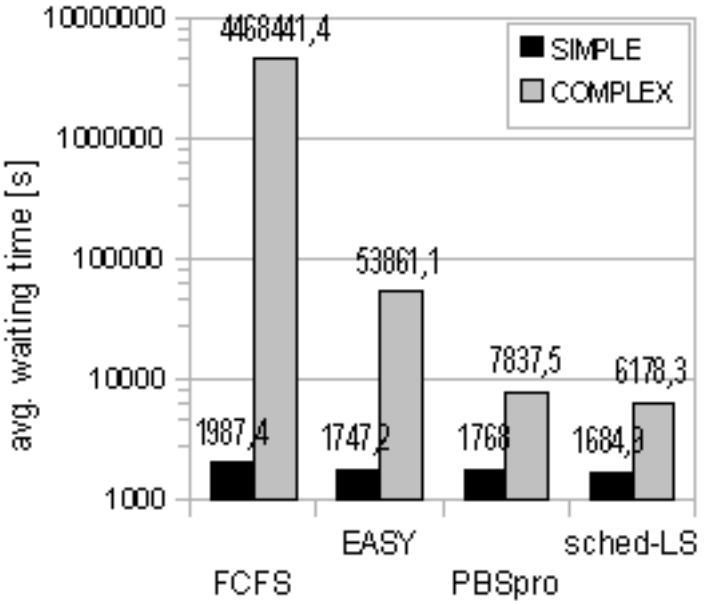
Výpadky strojů a specifické požadavky úloh

- V reálu:
  - Stroje přibývají a ubývají
  - Úlohy „umírají“
  - Stroje jsou různé
  - Uživatelé chtějí různé věci (specifické požadavky)
  - Jen některé stroje „umí“ některé věci
  - Administrátoři nepustí všechny všude
  - …
- **Bude to mít vliv na chování algoritmů?**



Vliv na výsledky

- Vliv je dramatický!
- Srovnejte **SIMPLE** vs. **COMPLEX** případy
- Rozdíly v algoritmech se mnohdy projeví až v případě použití komplexních dat
- Ta přitom nejsou běžně k dispozici… ☹



Závěr

- Plánování v Gridech je složité
  - Dynamika
  - Velký počet úloh
  - Přicházejí s různou frekvencí
  - Výpadky
  - Specifické požadavky
  - Simulační data jsou mnohdy nedostatečná
- Návrh algoritmů je složitý, chceme-li komplexnost i jednoduchost, kvalitu i rychlost, …