**Основи на паралелните системи (Concurrency Basics) - Резюме**

## 1. Въведение в паралелните системи

**Паралелността** е свойство на компютърните системи, при което **няколко изчислителни процеса се изпълняват едновременно и взаимодействат помежду си**. **Паралелността е синхронизиран достъп до споделени ресурс(и)**

**Основните предизвикателства включват:**

- Сложността на поведението на системата
- Координация между процесите, обмен на данни и управление на паметта.
- Минимизиране на времето за отговор и максимизиране на производителността.

## 2. Основни понятия и модели

**Споделени ресурси:**

- **Споделени променливи**: Променливи, които се четат и модифицират от различни процеси. След като процесът P запише стойност на променливата x, то процесът Q може да я прочете.
- **Споделени комуникационни канали**: каналът е среда, през която съобщенията се предават. Процесът P може да изпраща съобщения, които процесът Q да получава: закъснение на разпространението, буфериране на съобщенията, ред на съобщенията, загуба на информация, дублиране, грешки.

**Моделиране на паралелни системи:**

-Модел на споделена памет

- Модел на преминаване на съобщения (message-passing model).

Изборът зависи от: х-ките на системата; начинът за разсъждение; субективни преценки (вкус, предпочитание, умения).

**паралелното изпълнение** на **P || Q**

**interleaved** (преплетено) - При паралелно изпълнение, инструкциите на **P** и **Q** могат да се редуват в произволен ред.

Тагване (**Tagging**) се използва за разграничаване на действията на двата процеса при проследяване на събитията. Всеки процес е "тагиран" със свое име, за да е ясно кой процес изпълнява конкретна инструкция: P: x := 1; Q: x := x + 1

При **последователно изпълнение** резултатът е предсказуем и зависи от фиксирания ред на изпълнение. При **паралелно изпълнение** с **interleaving** резултатът зависи от точния ред на действията, което води до **недетерминизъм** (повече от един възможен изход). Точно поради това паралелните процеси изискват специални механизми за синхронизация, за да се избегнат неочаквани резултати.

**3. Синхронизация и атомарност**

**Атомарни операции**: Операции, които не могат да бъдат прекъснати или разделени; не може да има междинно състояние

**Формализмът** се използва, за да се опише точно и недвусмислено изпълнението на паралелни процеси, като се вземат предвид всички възможни редове на изпълнение (последователни или паралелни).

При работа с паралелни процеси, особено в контекста на споделени ресурси, като променливи, формализмът е необходим за:

**-Описване на процесите:** Всеки процес се представя с формални инструкции, които задават действията върху ресурсите. Процес **P**: x := 1

Процес **Q**: x := x + 1

**-Моделиране на паралелността:** Формализъмът представя изпълнението на процесите чрез:

**Последователност:** P; Q или Q; P (един процес следва друг).

**Паралелност:** P || Q (процесите се изпълняват паралелно, като инструкциите могат да се преплитат).

**Начини за комуникация:**

**-Модел Споделена памет** – 1/Споделена променлива ( или ресурс) 2/ Множество от споделени променливи

**- Message passing модел:**

**1/ Комуникационен канал (буфер ) -** представлява средство за обмен на съобщения между процеси, които могат да бъдат част от една или различни системи. Най-често този канал включва **буфер** – място за временно съхранение на данни. Атомарните операции в този контекст са действия, които се изпълняват неделимо, тоест те не могат да бъдат прекъснати от други процеси: Изпращане на съобщение (Send); Получаване на съобщение (Receive); Достъп до буфера

**2/ Атомарно разпространение (remote procedure call(RPC)) -** Remote Procedure Call (RPC) е техника, която позволява на един

процес (клиент) да извика функция или процедура, която се изпълнява на друг процес (сървър), разположен на отдалечена машина. Атомарното разпространение се отнася до осигуряване на неделимост на тези извиквания. Примери за атомарни действия в RPC: Изпращане на заявка; Изпълнение на процедура; Връщане на резултат

**Синхронизация:** Осигуряване на безопасен достъп до споделените ресурси:

**1/Модел Споделена памет**

**А)Взаимно изключване (mutual exclusion):** Предотвратява едновременното изпълнение на конфликтни действия. За да се предотвратят конфликти при едновременен достъп на множество процеси до споделен ресурс, се използват **синхронизиращи конструкции**. Целта е да се гарантира, че само един процес може да използва даден ресурс в даден момент. **Синхронизиращи конструкции:**

**-Семафори (semaphores):** Променливи, използвани за ограничаване на достъпа до ресурс чрез брояч. Например: wait() и signal().

**-Условни променливи (condition variables):** Използват се в комбинация с блокировки, за да се управлява състоянието на процесите, които чакат определено условие.

**-Монитори (monitors):** Абстракции, които комбинират взаимно изключване и управление на условията. Осигуряват по-структуриран подход към синхронизацията.

**-Други:** Спинлокове (spinlocks), бариери (barriers), атомарни опер-и.

Б) **Прочитане на последната записана стойност -** В системи със споделена памет е важно процесите винаги да имат достъп до **актуалната стойност** на споделена променлива.

Модел Споделена памет, моделиран с машина на състоянието: Трябва да се следи изпълнението на отделните паралелни примитивни процеси. За целта се въвежда:

-**програмен брояч (program counter) (pc**) за всеки примитивен процес. Той се прибавя към променливите, които дефинират състоянието на съответния примитив

-Начин за различаване на преходи между състоянията, който си приличат изисква таг (**to tag**) с името на процеса към всяко действие, чиято реализация променя състоянието.

**2/Message passing модел**

Координиране на изпращането и получаването на съобщения

• Видове:

- asynchronous : non-blocking sender (example?)

- synchronous : blocking sender and receiver (example?)

• Буфериране (Buffered message passing - channel capacity)

• Communicating Sequential Processes (CSP), a model of concurrent

systems is based on synchronous message passing (C.A.R. Hoare, 1985)

**4. Паралелно изпълнение**

**Последователно (sequential):** Един процес завършва преди друг да започне.

**Паралелно:** Два или повече процеса се изпълняват едновременно с потенциално преплитане (interleaving).

Проблеми като **състезателни условия (race conditions)** изискват специално внимание при проектиране.

## 5.Комбиниране на машини на състоянията (State Machines)

### 1) Моделират се поотделно всички примитивни процеси:

Всеки процес (P и Q) се моделира поотделно със състояния, входни и изходни променливи и преходи. Например:

$P = (S_P, I_P, A_P, d_P)$

$Q = (S_Q, I_Q, A_Q, d_Q)$

Тези процеси се комбинират заедно за да се създаде нова система, която обединява състоянията на двата процеса.

### 2) Трасета на комбинирани процеси (P || Q):

Комбинираната машина на състоянието **P || Q има трасета**, които могат да съвпадат както с **трасета на P,** така и **с трасета на Q**, но има и такива, които **не съвпадат с нито едно от тях**. Това означава, че системата P || Q не е просто обединение на трасетата на P и Q, а има **нови възможни преходи**:

**traces(P || Q) ≠ traces(P) ∩ traces(Q)**

**traces(P || Q) ≠ traces(P) ∪ traces(Q)**

### 3) Разделяне на променливите в локални и глобални:

Локалните променливи се отнасят само към един процес (например, само към P или Q) и се използват само от него.

Глобалните променливи са споделени между процесите и могат да се променят от всеки от тях.

**4/Прибавяне на програмния брояч (pcP) към локалните променливи:** Към всяка локална променлива на процес P се добавя програмния брояч $pc_P$, който показва текущото състояние на процеса.

**5/Дефиниране на състоянията на комбинираната машина на състоянието:** Комбинираната система P || Q се състои от **Състояния**:

$$\mathbf{S_{P||Q} = (locals(P) \times locals(Q) \times globals(P, Q)) \twoheadrightarrow Val}$$

**globals(P, Q) = vars(P)∩vars(Q)-**споделени променливи между P и Q

**locals(P) = vars(P) \ globals(P, Q) -**локални променливи на P

**locals(Q) = vars(Q) \ globals(P, Q) -**локални променливи на Q

Всеки процес има собствен набор от локални променливи и може да променя своите локални стойности и споделените глобални ст-сти.

**6) Начални състояния и действия:**

Началните състояния на комбинираната система са съставени от началните състояния на P и Q.

Действията се комбинират, като се обединяват действията на P и Q:

$A_{P||Q} = A_P \cup A_Q$

Забележка 1: Ако всяко действие е етикетирано с името на процеса, то действията на P и Q няма да се припокриват ($AP \cap AQ = \emptyset$).

Забележка 2: Ако искаме споделени действия, можем да използваме подхода за етикетиране, без да се налага да правим разграничение.

**7) Функция на преходите:** Преходите на комбинираната система включват стъпки, които може да бъдат изпълнени от процесите P или Q. Когато процес P извършва стъпка, неговият програмния брояч pc се увеличава, а локалните променливи на Q остават непроменени. Аналогично, когато процес Q извършва стъпка, неговият pc се увеличава, а локалните променливи на P не се променят.

**8) Единствен преход на комбинирана машина на състоянието с n процеси**: Комбинираната машина може да има преходи, които се извършват поотделно от всеки процес, като се обновяват състоянията на съответния процес, без да се променят състоянията на остан.п-си.

**6. Приложения:** Моделът на споделена памет е широко използван **в програмни езици,** поддържащи многопоточност, като C-Threads, Modula-2++, Modula-3. Прилага се в **транзакционни системи**, които изискват централизирани или глобално разпределени бази данни.

**Моделът комбинира МС, но не е композитен**, защото няма еквивалентост: **traces(P || Q) ≠traces(P) ∩ traces( Q )**

traces( P || Q ) ) ≠ traces( P ) U traces( Q )

Характеристиката (предикат) Inv , който е валиден и за P и за Q не задължително е валиден за паралелния процес P || Q

**Inv( P ) ^ Inv( Q ) не следва Inv( P || Q)** • Характеристика, която е валидна и за P || Q е **глобална инвариантa**