#### Lecture 8

 $\circ$ 

**Parallelization Strategies** 

Institute of Computer Science & Information Technology, Faculty of Management & Computer Sciences, The University of Agriculture, Peshawar, Pakistan.

#### Parallelization Strategies: Challenges

- Parallelization Strategies: Challenges<br>• Selecting the best parallelization strategy:<br>using proper parallelization strategy is crucial<br>in order to achieve the optimal performance arallelization Strategies: Challenges<br>Selecting the best parallelization strategy:<br>using proper parallelization strategy is crucial<br>in order to achieve the optimal performance,<br>such that the calculation takes the shortest arallelization Strategies: Challenges<br>Selecting the best parallelization strategy:<br>using proper parallelization strategy is crucial<br>in order to achieve the optimal performance,<br>such that the calculation takes the shortest arallelization Strategies: Challenges<br>Selecting the best parallelization strategy:<br>using proper parallelization strategy is crucial<br>in order to achieve the optimal performance,<br>such that the calculation takes the shortest<br> possible time. • Selecting the best parallelization strategy:<br>using proper parallelization strategy is crucial<br>in order to achieve the optimal performance,<br>such that the calculation takes the shortest<br>possible time.<br>• It is very difficul Selecting the best parallelization strategy.<br>
using proper parallelization strategy is crucial<br>
in order to achieve the optimal performance,<br>
such that the calculation takes the shortest<br>
possible time.<br>
It is very difficu
- depends on the calculation of a<br>team of the position of alternation is a team of that the calculation takes the shortest<br>possible time.<br>It is very difficult to give a universal method<br>for finding the optimal strategy, sinc In order to achieve the optimal perform<br>such that the calculation takes the shop<br>possible time.<br>It is very difficult to give a universal metric finding the optimal strategy, sin<br>depends not only on the platform, but al<br>the



#### Parallelization Strategies

- Parallelization Strategies define how to implement parallelization opportunities.
- How much levels?
- How much granular?

#### **Granularity**

- Parallelization granularity refers to the size of parallel entities at which we can divide an application.
- Granularity is concerned with depth or level of details.



"Specific" is concerned with scape whereas







Understand the ideo/concept

Able to describe the idea/concept with some detail

Able to describe and explain the idea/concept with some detail

Able to describe and explain the idea/concept with full detail

Fine Grain

More granular



#### **Granularity**

- **Granularity**<br>• For both levels, we may define mainly two granularities: granularities: Granularity<br>
• For both levels, we may definged<br>
Fine Grain<br>
1. Fine Grain<br>
2. Coarse Grain Granularity<br>
• For both levels, we may define<br>
granularities:<br>
1. Fine Grain<br>
2. Coarse Grain
- 
- 
- For both levels, we may define mainly two<br>granularities:<br>1. Fine Grain<br>2. Coarse Grain<br>• Coarse grained materials or systems have fewer,<br>larger components than <u>fine-grained</u> systems. For both levels, we may define mainly two<br>granularities:<br>Fine-Grain<br>Coarse-Grain<br>Coarse grained materials or systems have fewer,<br>larger-components than <u>fine-grained</u> systems.<br>...
- 1. Fine Grain<br>
2. Coarse Grain<br>
 <u>Coarse grained</u> materials or systems have fewer,<br>
larger components than <u>fine-grained</u> systems.<br>
 The concepts granularity, coarseness, and<br>
<u>fineness</u> are relative; are used when we ar Coarse grained materials or systems have fewer,<br>
larger components than <u>fine-grained</u> systems.<br>
The concepts granularity, coarseness, and<br>
fineness are relative; are used when we are<br>
decomposing some system. Coarse grained materials or systems have few<br>larger components than <u>fine-grained</u> systems.<br>The concepts granularity, coarseness, a<br>fineness are relative; are used when we a<br>decomposing some system.



#### Coarse Vs. Fine Grain

- **Coarse Vs. Fine Grain**<br>• Fine-grained parallelism means individual task<br>are relatively small in terms of <u>code size</u> and<br>execution time Coarse Vs. Fine Grain<br>Fine-grained parallelism means individual task<br>are relatively small in terms of <u>code size a</u>nd<br>execution time. Coarse Vs. Fine Grain<br>
• Fine-grained parallelism means individual task<br>
are relatively small in terms of <u>code size</u> and<br>
<u>execution time</u>.<br>
• The data is transferred among processors<br>
<u>frequently</u> in amounts of one or a
- frequently in amounts of one or a few Fine-grained parallelism means incordent and the means of <u>conditation time</u>.<br>The data is transferred among frequently in amounts of one memory words. execution time.<br>
• The data is transferred among processors<br>
<u>frequently</u> in amounts of one or a few<br>
memory words.<br>
• Coarse-grained is the opposite: data is<br>
communicated infrequently, after larger<br>
amounts of computatio
- The data is transferred among processors<br>
<u>frequently</u> in amounts of one or a few<br>
memory words.<br>
Coarse-grained is the opposite: data is<br>
communicated infrequently, after larger<br>
amounts of <u>computation</u>. • The data is transferred among processor<br>frequently in amounts of one or a fev<br>memory words.<br>• Coarse-grained is the opposite: data i<br>communicated infrequently, after large<br>amounts of <u>computation</u>.<br><sup>Dr. Muhammad Asim, IC</sup>

# Coarse Vs. Fine Grain (1)

- Coarse Vs. Fine Grain (1)<br>• The finer the granularity, the greater the potential for parallelism and hence speed-up, but the greater the overheads of synchronization and Coarse Vs. Fine Grain (1)<br>The finer the granularity, the greater the<br>potential for parallelism and hence speed-up, but<br>the greater the <u>overheads</u> of <u>synchronization</u> and<br>communication. **Coarse Vs. Fine Grain (1)**<br>The finer the granularity, the greater the<br>potential for parallelism and hence speed-up, but<br>the greater the <u>overheads</u> of <u>synchronization</u> and<br>communication. communication. The finer the granularity, the greater the<br>potential for parallelism and hence speed-up, but<br>the greater the <u>overheads</u> of <u>synchronization</u> and<br>communication.<br>On the other side, if the granularity is too<br>coarse, the perf
- On the other side, if the granularity is too imbalance. In the greater the <u>overneads</u> of <u>synchronization</u> and<br> **Communication**.<br> **•** On the other side, if the granularity is too<br>
coarse, the performance can suffer from <u>load</u><br>
<u>Inhalance</u>.<br>
• In order to attain the best <u>para</u> communication.<br>
On the other side, if the granularity is too<br>
coarse, the performance can suffer from <u>load</u><br>
imbalance.<br>
In order to attain the best <u>parallel performance</u>,<br>
the best <u>balance</u> between load and<br>
<u>communica</u> On the other side, if the granularity is too coarse, the performance can suffer from <u>load imbalance.</u><br>In order to attain the best <u>parallel performance</u>, the best <u>balance</u> between load and communication overhead needs to
- 





- 
- 
- Coarse Vs. Fine Grain (2)<br>1. Task-level parallelization<br>2. Data-level parallelization<br>• For both levels we may define mainly two<br>granularities: granularities: 1. Fine text parameterization<br>2. Data-level parallelization<br>• For both levels we may defir<br>granularities:<br>1. Fine grain<br>2. Coarse grain 2. Data force paramements.<br>
• For both levels we may define is<br>
granularities:<br>
1. Fine grain<br>
2. Coarse grain
- 
- 



#### **Task-level Granularity**

- Task-level Granularity<br>• Task-level granularity is directly related to the<br>program decomposition into <u>independent</u> Task-level Granularity<br>Task-level granularity is directly related to the<br>program decomposition into <u>independent</u><br>tasks. tasks. • Task-level graindarity is directly related program decomposition into indefinite.<br>
tasks.<br>
• It has two types:<br>
1. Fine grain tasking<br>
2. Coarse grain tasking Examples and tasks.<br>
1. Fine grain tasking<br>
2. Coarse grain tasking<br>
2. Coarse grain tasking
- It has two types:
- 
- 

#### Fine Grain Tasking

- Fine Grain Tasking<br>• Fine-grain tasking consists of dividing the<br>program in fundamental separate tasks and<br>each single task is parallelized apart Fine Grain Tasking<br>Fine-grain tasking consists of dividing the<br>program in fundamental separate tasks and<br>each single task is parallelized apart<br>(separately). Fine Grain Tasking<br>Fine-grain tasking consists of dividing the<br>program in fundamental separate tasks and<br>each single task is parallelized apart<br>(separately). (separately). • Fine-grain tasking consists or dividing the<br>program in fundamental separate tasks and<br>each single task is parallelized apart<br>(separately).<br>• This process is called fission.<br>• The benefit of the fine grain parallelization
- This process is called fission.
- program in fundamental separate tasks and<br>each single task is parallelized apart<br>(separately).<br>This process is called fission.<br>The benefit of the fine grain parallelization<br>strategy is the high reusability, since each task each single task is paraileilzed apart<br>(separately).<br>This process is called fission.<br>The benefit of the fine grain parallelization<br>strategy is the high reusability, since each task<br>may be found in more than one algorithm<br>w (separately).<br>This process is called fission.<br>The benefit of the fine grain parallelization<br>strategy is the high reusability, since each task<br>may be found in more than one algorithm<br>which is the case of most image processi algorithms.



# Fine Grain Tasking(1)

- Fine Grain Tasking(1)<br>• This means that a parallel version of this operation can be reused more than once in Fine Grain Tasking(1)<br>This means that a parallel version of this<br>operation can be reused more than once in<br>different applications without any modification Fine Grain Tasking(1)<br>This means that a parallel version of this<br>operation can be reused more than once in<br>different applications without any modification<br>to ensure high portability among different Fine Grain Tasking(1)<br>This means that a parallel version of this<br>operation can be reused more than once in<br>different applications without any modification<br>to ensure high portability among different<br>applications. applications. • This means that a parallel version of this<br>operation can be reused more than once in<br>different applications without any modification<br>to ensure high portability among different<br>applications.<br>• However, this strategy may s operation can be reused more than once in<br>different applications without any modification<br>to ensure high portability among different<br>applications.<br>• However, this strategy may suffer from:<br>1. Overhead introduced by success
- 
- launches. 2. Poor temporal data locality.
- 



### Coarse Grain Tasking

- **Coarse Grain Tasking<br>• Coarse-grain tasking consists in packing a**<br>• sequence of tasks into a macro task. Coarse Grain Tasking<br>Coarse-grain tasking consists in packing a<br>sequence of tasks into a macro task. Coarse Grain Tasking<br>
• Coarse-grain tasking consists in packing a<br>
sequence of tasks into a macro task.<br>
• This process is called fusion.
- 
- Coarse-grain tasking consists in packing a<br>sequence of tasks into a macro task.<br>• This process is called fusion.<br>• Each macro task is assigned to a single thread<br>which processes a part of data array. Coarse-gram tasking consists in packing a<br>sequence of tasks into a macro task.<br>This process is called fusion.<br>Each macro task is assigned to a single thread<br>which processes a part of data array.

# Coarse Grain Tasking (1)

- **Coarse Grain Tasking (1)**<br>• The implementation of this parallelization<br>strategy needs additional programming effort<br>to manage dependencies between neighbors Coarse Grain Tasking (1)<br>The implementation of this parallelization<br>strategy needs additional programming effort<br>to manage dependencies between neighbors<br>data in order to minimize the synchronization **Coarse Grain Tasking (1)**<br>The implementation of this parallelization<br>strategy needs additional programming effort<br>to manage dependencies between neighbors<br>data in order to minimize the synchronization<br>barriers and data co Coarse Grain Tasking (1)<br>The implementation of this parallelization<br>strategy needs additional programming effort<br>to manage dependencies between neighbors<br>data in order to minimize the synchronization<br>barriers and data comm **Coarse Grain Tasking (1)**<br>The implementation of this parallelizat<br>strategy needs additional programming eff<br>to manage dependencies between neighb<br>data in order to minimize the synchronizat<br>barriers and data communication. • The implementation of this parallelization<br>strategy needs additional programming effort<br>to manage dependencies between neighbors<br>data in order to minimize the synchronization<br>barriers and data communication.<br>When the imp The implementation of this paramerization<br>strategy needs additional programming effort<br>to manage dependencies between neighbors<br>data in order to minimize the synchronization<br>barriers and data communication.<br>When the implem
- strategy needs additional programming enort<br>to manage dependencies between neighbors<br>data in order to minimize the synchronization<br>barriers and data communication.<br>When the implementation of the coarse-grain<br>strategy is op Lo manage dependencies between helghbors<br>data in order to minimize the synchronization<br>barriers and data communication.<br>When the implementation of the coarse-grain<br>strategy is optimized, runtime performances<br>may be improve barriers and data communication.<br>
When the implementation of the coarse-grain<br>
strategy is optimized, runtime performances<br>
may be improved by increasing temporal data<br>
locality and by avoiding overhead cause by<br>
threads l

# Coarse Grain Tasking (2)

Coarse Grain Tasking (2)<br>• Temporal locality refers to the reuse of<br>specific data and/or resources within a Coarse Grain Tasking (2)<br>Temporal locality refers to the reuse of<br>specific data and/or resources within a<br>relatively small time duration. Coarse Grain Tasking (2)<br>Temporal locality refers to the reuse of<br>specific data and/or resources within a<br>relatively small time duration.



#### Data-Level Granularity

- Data-Level Granularity<br>• Data level granularity defines the degree of<br>decomposition of initial data into data Data-Level Granularity<br>Data level granularity defines the degree of<br>decomposition of initial data into data<br>subsets. subsets. • Data level grafitularity defines the decomposition of initial data in<br>subsets.<br>• It has two types:<br>1. Fine-grain Tiling<br>2. Coarse-grain Tiling
- It has two types:
- 
- 2. Coarse-grain Tiling

# Fine-grain Tiling

- Fine-grain Tiling<br>• Fine-grain Tiling consists in decomposing the<br>data into small subsets (small tiles in the case<br>of image processing) Fine-grain Tiling<br>Fine-grain Tiling consists in decomposing the<br>data into small subsets (small tiles in the case<br>of image processing). Fine-grain Tiling<br>Fine-grain Tiling consists in decor<br>data into small subsets (small tiles<br>of image processing). Fine-grain Tiling<br>
• Fine-grain Tiling consists in decomposing the<br>
data into small subsets (small tiles in the case<br>
of image processing).<br>
• These small tiles are assigned to small groups<br>
of threads.
- of threads.
- data into small subsets (small tiles in the case<br>of image processing).<br><br>• These small tiles are assigned to small groups<br>of threads.<br>• This strategy exposes a high-degree of<br>concurrency and takes advantage of<br>architectures of image processing).<br>These small tiles are assigned to small groups<br>of threads.<br>This strategy exposes a high-degree of<br>concurrency and takes advantage of<br>architectures supporting a huge number of<br>threads These small tiles are assigned to small groups<br>of threads.<br>This strategy exposes a high-degree of<br>concurrency and takes advantage of<br>architectures supporting a huge number of<br>threads. threads.



- Fine-grain Tiling (1)<br>• In addition, this strategy is usually not constrained by the hardware resources<br>limitations. Fine-grain Tiling (1)<br>In addition, this strategy is usually not<br>constrained by the hardware resources<br>limitations. limitations.
- Fine-grain Tiling (1)<br>
 In addition, this strategy is usually not<br>
constrained by the hardware resources<br>
limitations.<br>
 However, it suffers from a significant overhead<br>
in processing replicated data at borders to<br>
handl in processing replicated data at borders to handle boundary dependencies.
- Four addition, this strategy is usually not<br>constrained by the hardware resources<br>limitations.<br>• However, it suffers from a significant overhead<br>in processing replicated data at borders to<br>handle boundary dependencies.<br>• B by the hardware resources<br>limitations.<br>However, it suffers from a significant overhead<br>in processing replicated data at borders to<br>handle boundary dependencies.<br>Boundaries are created as per purposes, each<br>boundary must cr However, it suffers from a significant overhead<br>in processing replicated data at borders to<br>handle boundary dependencies.<br>Boundaries are created as per purposes, each<br>boundary must create objects related to a<br>single area o etc.

# **Coarse-grain Tiling**

- **Coarse-grain Tiling<br>• Coarse-grain Tiling consists in decomposing<br>data into large data subsets and involving large<br>groups of threads Coarse-grain Tiling**<br>Coarse-grain Tiling consists in decomposing<br>data into large data subsets and involving large<br>groups of threads. **Coarse-grain Tiling**<br>Coarse-grain Tiling consists in de<br>data into large data subsets and inve<br>groups of threads.
- This strategy reduces the data replication at borders and offers high space data locality.
- data into large data subsets and involving large<br>groups of threads.<br>• This strategy reduces the data replication at<br>borders and offers high space data locality.<br>• However, large tiles may not fit well with<br>available resour groups of threads.<br>This strategy reduces the data replication at<br>borders and offers high space data locality.<br>However, large tiles may not fit well with<br>available resources, cache or local memory<br>size, which may degrade pe This strategy reduces the data replication at<br>borders and offers high space data locality.<br>However, large tiles may not fit well with<br>available resources, cache or local memory<br>size, which may degrade performance.

#### Types of Parallelization

- Types of Parallelization<br>• Types of parallelism or also known as<br>parallelism models, define the way to organize<br>independent workflows Types of Parallelization<br>Types of parallelism or also known as<br>parallelism models, define the way to organize<br>independent workflows. Types of Parallelization<br>Types of parallelism or also kno<br>parallelism models, define the way to dindependent workflows. Types of Parallelization<br>
• Types of parallelism or also known as<br>
parallelism models, define the way to organize<br>
independent workflows.<br>
• We can distinguish three main types of<br>
parallelism: Suite of the Mayarallelism models, define the Wayaraldependent workflows.<br>
• We can distinguish three main parallelism:<br>
1. Data parallelism<br>
2. Task parallelism
- parallelism: independent workflows.<br>• We can distinguish three mair<br>parallelism:<br>1. Data parallelism<br>2. Task parallelism<br>3. Pipeline parallelism • We can distinguish three main t<br>1. Data parallelism<br>2. Task parallelism<br>3. Pipeline parallelism
- 
- 
- 



#### Data Parallelism

- **Data Parallelism**<br>• Data parallelism refers to work units<br>executing the same operations on a set of<br>data **Example 18 Data Parallelism**<br>
Data parallelism refers to work units<br>
executing the same operations on a set of<br>
data. data. Data parallelism refers to work un<br>executing the same operations on a set<br>data.<br>The data is typically organized into a comm<br>structure such as arrays.
- The data is typically organized into a common
- Executing the same operations on a set of<br>
data.<br>
The data is typically organized into a common<br>
structure such as arrays.<br>
<br>
Each work unit performs the same operations<br>
as other work units but on different elements<br>
of t data.<br>The data is typically organized into a common<br>structure such as arrays.<br>Each work unit performs the same operations<br>as other work units but on different elements<br>of the data structure. The data is typically organized into a<br>structure such as arrays.<br>Each work unit performs the same o<br>as other work units but on different<br>of the data structure.

# Data Parallelism (1)

- **Data Parallelism (1)**<br>• The first concern of this parallelism type is<br>how to distribute data on work units while Data Parallelism (1)<br>The first concern of this parallelism type is<br>how to distribute data on work units while<br>keeping them independent. Data Parallelism (1)<br>The first concern of this parallelism ty<br>how to distribute data on work units<br>keeping them independent. The first concern of this parallelism type is<br>how to distribute data on work units while<br>keeping them independent.<br>Data parallelism is generally easier to exploit<br>due to the simpler computational model<br>involved.
- Data parallelism is generally easier to exploit involved.

#### Task Parallelism

- Task Parallelism<br>• Task parallelism is known as functional<br>parallelism or control parallelism. Task Parallelism<br>Task parallelism is known as functional<br>parallelism or control parallelism.
- Task Parallelism<br>
 Task parallelism is known as functional<br>
parallelism or control parallelism.<br>
 This type of parallelism considers the case<br>
when work units are executing on different<br>
control flow paths when work units are executing on different Task parallelism is known as funderallelism or control parallelism.<br>This type of parallelism considers to<br>when work units are executing on exentrol flow paths. • This type of parallelism considers the case<br>• This type of parallelism considers the case<br>when work units are executing on different<br>control flow paths.<br>• Work units may execute different operations<br>on either the same da This type of parallelism considers the case<br>when work units are executing on different<br>control flow paths.<br>Work units may execute different operations<br>on either the same data or different data.
- 

# Task Parallelism (1)

- Task Parallelism (1)<br>• In task parallelism, work units can be known<br>at the beginning of execution or can be Task Parallelism (1)<br>In task parallelism, work units can be known<br>at the beginning of execution or can be<br>generated at runtime. Task Parallelism (1)<br>In task parallelism, work units can be<br>at the beginning of execution or<br>generated at runtime.
- Task parallelism could be expressed in the In task parallelism, work units can be known<br>at the beginning of execution or can be<br>generated at runtime.<br>Task parallelism could be expressed in the<br>GPU context in two ways. In a multi-GPU<br>environment, each task could be In task parallelism, work units can be known<br>at the beginning of execution or can be<br>generated at runtime.<br>Task parallelism could be expressed in the<br>GPU context in two ways. In a multi-GPU<br>environment, each task could be at the beginning of execution of<br>generated at runtime.<br>Task parallelism could be express<br>GPU context in two ways. In a r<br>environment, each task could be pro<br>a separate GPU.

# Task Parallelism (2)

Task Parallelism (2)<br>
In a single GPU environment, the task<br>
parallelism is expressed as independent kernel Task Parallelism (2)<br>In a single GPU environment, the task<br>parallelism is expressed as independent kernel<br>queues called respectively streams and Task Parallelism (2)<br>In a single GPU environment, the task<br>parallelism is expressed as independent kernel<br>queues called respectively streams and<br>Command queues in CUDA (Compute Task Parallelism (2)<br>In a single GPU environment, the task<br>parallelism is expressed as independent kernel<br>queues called respectively streams and<br>Command queues in CUDA (Compute<br>Unified Device Architecture) and OpenCL. Unified Device Architecture) and OpenCL. • In a single GPU environment, the task<br>parallelism is expressed as independent kernel<br>queues called respectively streams and<br>Command queues in CUDA (Compute<br>Unified Device Architecture) and OpenCL.<br>• Kernel queues are exe parallelism is expressed as independent kernel<br>queues called respectively streams and<br>Command queues in CUDA (Compute<br>Unified Device Architecture) and OpenCL.<br>Kernel queues are executed concurrently<br>where each kernel queue

queues called respectively streams and<br>Command queues in CUDA (Compute<br>Unified Device Architecture) and OpenCL.<br>Kernel queues are executed concurrently<br>where each kernel queue performs its<br>workload in a data-parallel fashi

#### Pipeline Parallelism

- Pipeline Parallelism<br>• Pipeline parallelism is also known as temporal parallelism. parallelism.
- **Pipeline Parallelism**<br>• Pipeline parallelism is also known as temporal<br>parallelism.<br>• This type of parallelism is applied to chains of<br>producers are consumers that are directly<br>connected. **Pipeline Parallelism**<br>Pipeline parallelism is also known as temporal<br>parallelism.<br>This type of parallelism is applied to chains of<br>producers are consumers that are directly<br>connected. connected. • Pipeline paralielism is also known as temporal<br>parallelism.<br>• This type of parallelism is applied to chains of<br>producers are consumers that are directly<br>connected.<br>• Each task is divided in a number of successive<br>phases. • This type or parallelism is applied to chains or<br>producers are consumers that are directly<br>connected.<br>• Each task is divided in a number of successive<br>phases.<br>• The result of each work unit is delivered to the<br>next for p
- phases. producers are consumers that<br>connected.<br>Each task is divided in a number<br>phases.<br>The result of each work unit is del<br>next for processing.
- 

# Pipeline Parallelism (1)

- **Pipeline Parallelism (1)**<br>• At the same time, the producer work unit starts<br>to process a give phase of a new task. Pipeline Parallelism (1)<br>At the same time, the producer work unit starts<br>to process a give phase of a new task.
- **Pipeline Parallelism (1)**<br>• At the same time, the producer work unit starts<br>to process a give phase of a new task.<br>• Compared to data parallelism, this approach<br>offers reduced latency, reduced buffering and<br>good locality. offers reduced latency, reduced buffering and At the same time, the producer<br>to process a give phase of a new<br>Compared to data parallelism<br>offers reduced latency, reduce<br>good locality. to process a give phase of a new task.<br>
• Compared to data parallelism, this approach<br>
offers reduced latency, reduced buffering and<br>
good locality.<br>
• However, this form of pipelining introduces extra<br>
synchronization, as Compared to data parallelism, this approach<br>offers reduced latency, reduced buffering and<br>good locality.<br>However, this form of pipelining introduces extra<br>synchronization, as producers and consumers<br>must stay tightly coupl Compared to data parallelism, this approach<br>offers reduced latency, reduced buffering and<br>good locality.<br>However, this form of pipelining introduces extra<br>synchronization, as producers and consumers<br>must stay tightly coupl
-