A Client Side Scheduler for UNICORE Based on HiLA

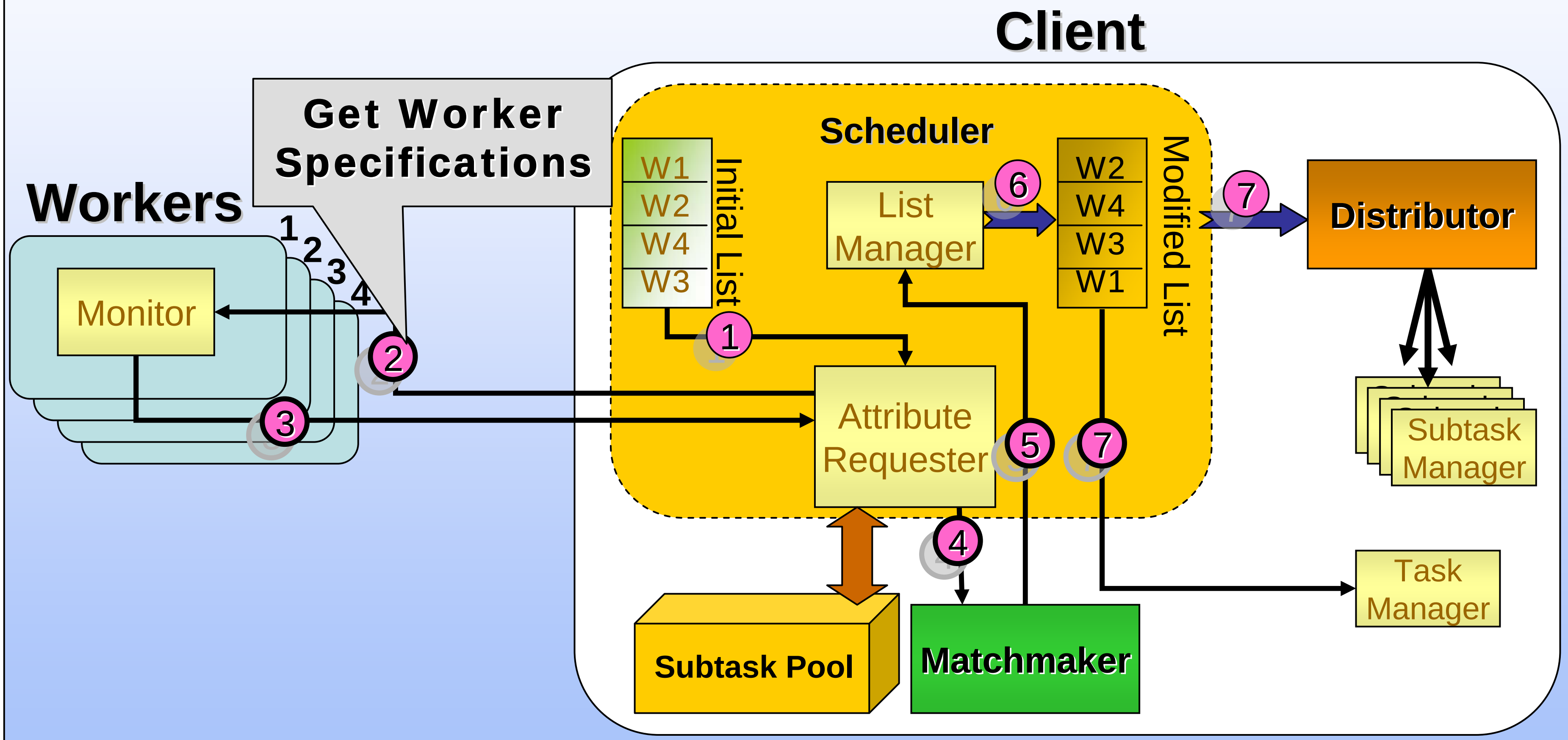
Abdulrahman Azab
University of Stavanger
abdulrahman.azab@uis.no

Abstract

The scheduling problem is specified by a set of workers, a set of tasks, an optimality criterion, environmental specifications, and by other constraints. The goal of a scheduling policy is to find an optimal schedule in the environment and to satisfy all constraints. The two main scheduling techniques are: centralized and decentralized. The main drawback of centralized scheduling is the central failure, which makes it non-practical for implementation in P2P and non-dependable environments. Decentralized scheduling is to locally schedule jobs to suitable workers from the client node. In this work, we propose a client side scheduler and Grid client for UNICORE based on HiLA API. The scheduling process is composed of two main steps: 1) gathering the resource information about TSs from the associated registry, and 2) assign submitted jobs to suitable TSs through adaptive matchmaking.

Scheduling Mechanism

Decentralized Scheduling scenario



Scheduling Mechanism

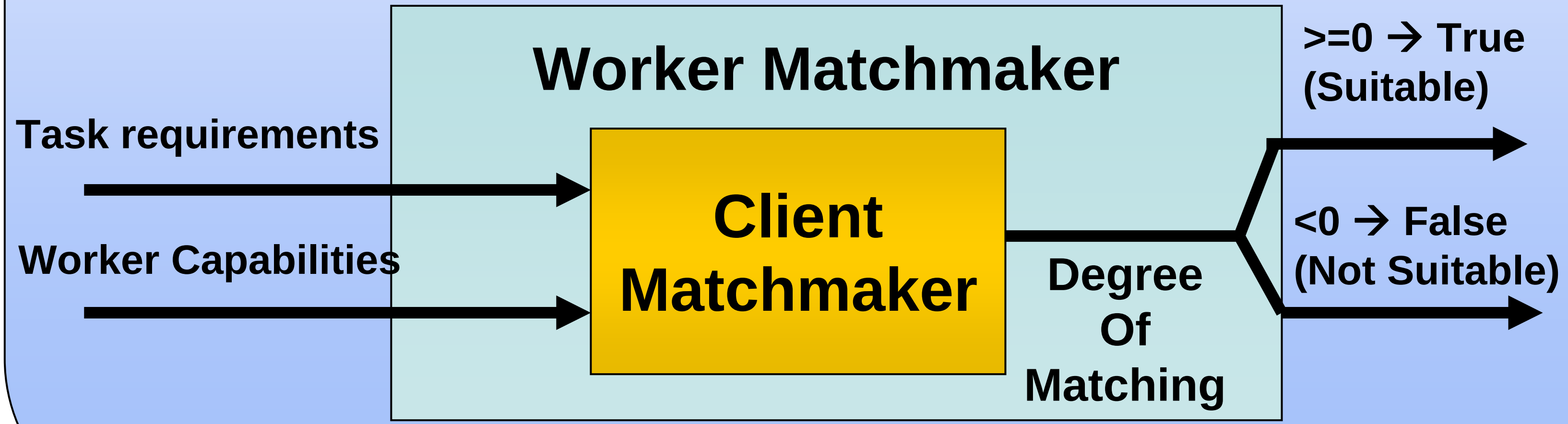
Matchmaking

Task requirements

Worker Capabilities

Client Matchmaker

Degree of Matching



Worker Matchmaker

Task requirements

Worker Capabilities

Client Matchmaker

Degree Of Matching

>=0 → True (Suitable)

<0 → False (Not Suitable)

Scheduling Mechanism

Matchmaking

Worker Capabilities:

Worker capabilities are calculated within a collection of time units {TU(i)→TU(i + NTU(T)) }, as follows:

Available CPU (VC) [MHz].

$$\mu_{ov}(w, VC, i, d, NTU(T)), \sigma_{ov}(w, VC, i, d, NTU(T))$$

Available Memory (VM) [MBytes]

$$\mu_{ov}(w, VM, i, d, NTU(T)), \sigma_{ov}(w, VM, i, d, NTU(T))$$

Number of Failures (NF)

$$NF_{ov}(w, i, d, NTU(T)) = Max(NF(w, j, d))$$

Scheduling Mechanism

Matchmaking

Task Requirements:

Task resource requirements are calculated from the resource usage of previous executions

CPU Cycles (CC (T)) [MCycles]

UC (T, w, e) = % Processor time of (w) used by (T) at execution (e)

CPU Time of (T,e) = UC(T,w,e) * Total execution time of (T) [sec]

CC(T, e) = CPU Time of (T,e) [sec] * CPU Speed of (w) [MHz]

$$m(T, CC) = Median(CC(T, e)) \quad \{e = 1, 2, 3, \ldots, E_f\}$$

Used Memory (UM (T)) [MBytes]

UM(T) = (Input data of (T) [MBytes] + Output data of (T) [MBytes] + Intermediate data during execution of (T) [MBytes]) [MBytes]

Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

The fuzzy matchmaking approach is implemented based on **Takagi-Sugeno** fuzzy model.

The following steps will construct the fuzzy inference process:

1. Fuzzification of Inputs.
2. Applying Fuzzy Operator.
3. Applying Implication Method.
4. Defuzzification.

Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

1. Fuzzification of Inputs

Each available worker (w) within the collection of **TUs {TU(i)→TU(i + NTU(T)) }** will have a **separate fuzzy** set represented by **two membership functions**. An input membership function is included for each parameter concerning **worker capabilities**.

The input membership functions can be described as follows:

$$Free\_CPU(x,w,i,d,NTU(T)) = \exp - \left(\frac{x - \mu_{ov}(w,VC,i,d,NTU(T))}{\sigma_{ov}(w,VC,i,d,NTU(T))} \right)$$

$$Free\_Memory(x,w,i,d,NTU(T)) = \exp - \left(\frac{x - \mu_{ov}(w,VM,i,d,NTU(T))}{\sigma_{ov}(w,VM,i,d,NTU(T))} \right)$$

Scheduling Mechanism

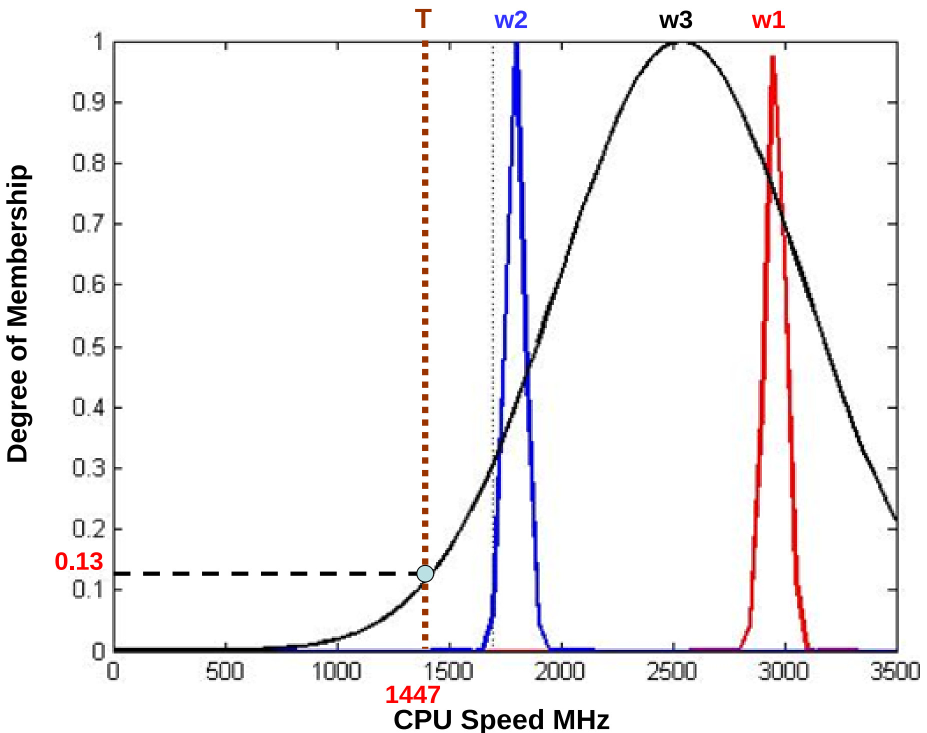
Fuzzy Matchmaking approach (FMA)

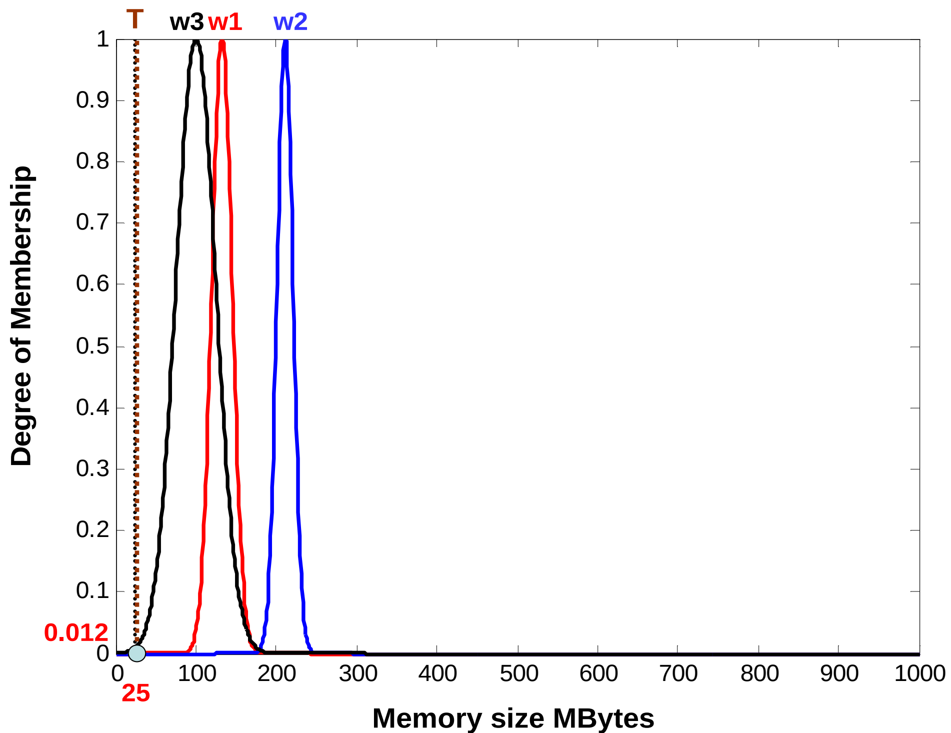
1. Fuzzification of Inputs

The input values to the fuzzification process can be described as follows:

$$\mathrm{Re}\,quired\_CPU(T)=\frac{m(T,CC)}{NTU(T)\times TU}$$

$$\mathrm{Re}\,quired\_Memory(T)=UM(T)$$





Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

2. Applying Fuzzy Operator

A separate rule will be created for each worker included in the matchmaking

If Required_CPU(T) Is Free_CPU(wj)
AND Required_Memory(T) Is Free_Memory(wj)
THEN Suitable_Worker(T) = ID(wj)

j = 1,2,…,N

N: number of workers

The rule weight is specified as a function of the number of failures:

$$RW(w,i,d,NTU(T)) = \frac{1}{NF_{ov}(w,i,d,NTU(T))}$$

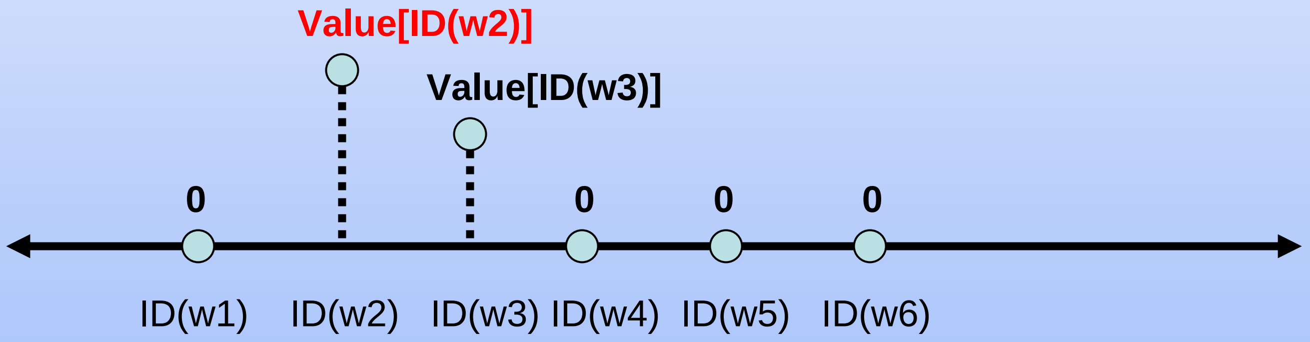
Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

3. Applying Implication Method

The consequent of a rule is an output fuzzy set represented by an output membership function.

The output membership function associated with each fuzzy set will be in the form of a unique identifier of the associated worker **[ID(wj) j = 1, 2, 3,…, N].**

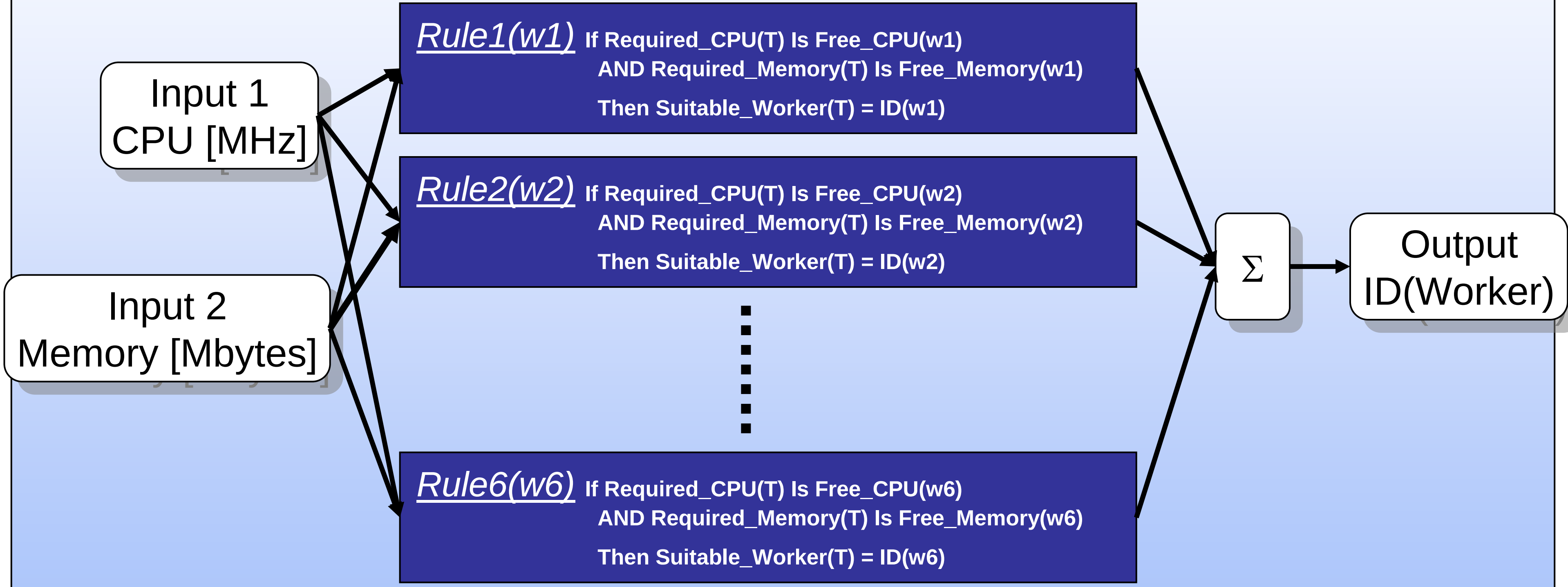


4. Defuzzification

Output = MAX (value [ID(w1)], value [ID(w2)],… , value [ID(wn)])

Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)



Input 1
CPU [MHz]

Input 2
Memory [Mbytes]

Rule1(w1) **If Required_CPU(T) Is Free_CPU(w1)**
AND Required_Memory(T) Is Free_Memory(w1)
Then Suitable_Worker(T) = ID(w1)

Rule2(w2) **If Required_CPU(T) Is Free_CPU(w2)**
AND Required_Memory(T) Is Free_Memory(w2)
Then Suitable_Worker(T) = ID(w2)

Rule6(w6) **If Required_CPU(T) Is Free_CPU(w6)**
AND Required_Memory(T) Is Free_Memory(w6)
Then Suitable_Worker(T) = ID(w6)

Σ

Output
ID(Worker)

Scheduling Mechanism

Simplified Fuzzy Matchmaking approach (SFMA)

Useful for use on the **Internet** where workers are PCs and **ruffling consumption level** of worker machines is expected.

Efficient for scheduling **short running** tasks.

Input membership functions:

$$Free\_CPU(x,w,i,d)=\begin{cases}\dfrac{x}{\mu_{cur}(w,VC,i,d)} & x\le\mu_{cur}(w,VC,i)\\[2ex] 0 & x>\mu_{cur}(w,VC,i)\end{cases}$$

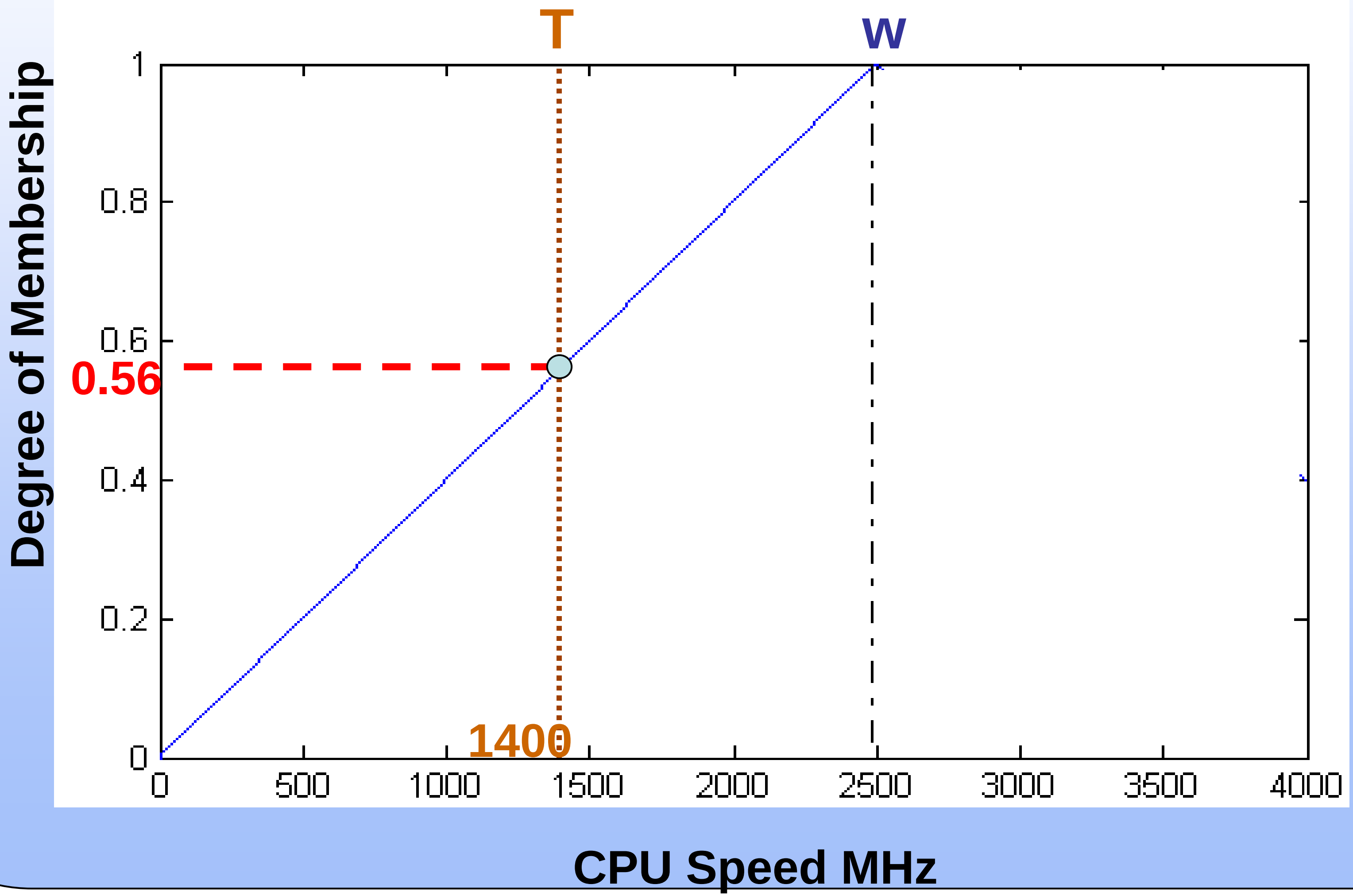
$$Free\_Memory(x,w,i,d)=\begin{cases}\dfrac{x}{\mu_{cur}(w,VM,i,d)} & x\le\mu_{cur}(w,VM,i)\\[2ex] 0 & x>\mu_{cur}(w,VM,i)\end{cases}$$

$$\mu_{cur}(w,VC,i,d)=\mu(w,VC,i-1,d)$$

$$\mu_{cur}(w,VM,i,d)=\mu(w,VM,i-1,d)$$

$$Free\_CPU(x,w,i)=\begin{cases}\dfrac{x}{2500} & x\le 2500\\ 0 & x>2500\end{cases}$$

$$Required\_CPU(T)=1400$$



Performance Evaluation

Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)

1. Parallel Execution Scheduling performance for 2400x2400 matrix size

Number Of subtasks	Scheduling Mechanism	Subtask index	%CPU utilization	%Memory utilization	Execution time (Seconds)	NTU(T) (Seconds)
2	SFMA	1	99.07	86.3	631	900
		2	94.4	60.1		
	Traditional	1	52.1	58.4	794	900
		2	96.3	81.8		
3	SFMA	1	98.6	54.3	440	600
		2	99.3	54.8		
		3	99.1	54.2		
	Traditional	1	99.7	88.7	778	600
		2	99.2	88.6		
		3	52.4	58.2		

Performance Evaluation

Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)

1. Parallel Execution Scheduling performance for 2400x2400 matrix size

Number Of subtasks	Scheduling Mechanism	Subtask index	%CPU utilization	%Memory utilization	Execution time (Seconds)	NTU(T) (Seconds)
4	SFMA	1	99.2	87.55	377.8	500
		2	98.4	87.3		
		3	97	88.1		
		4	60.8	98.9		
	Traditional	1	96.5	86.54	575	500
		2	99.5	87.4		
		3	98	87.2		
		4	59.3	71.5		

Performance Evaluation

Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)

1. Parallel Execution Scheduling performance for 2400x2400 matrix size

Number Of subtasks	Scheduling Mechanism	Subtask index	%CPU utilization	%Memory utilization	Execution time (Seconds)	NTU(T) (Seconds)
5	SFMA	1	99.33	55.5	354	400
		2	98.9	88.7		
		3	79.5	87.4		
		4	64.3	76.7		
		5	50.9	56.5		
	Traditional	1	58.6	72.3	502	400
		2	99.8	85.7		
		3	51.5	55.9		
		4	98.5	87		
		5	92.7	87.6		

Performance Evaluation

Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)

1. Parallel Execution Scheduling performance for 2400x2400 matrix size

Number Of subtasks	Scheduling Mechanism	Subtask index	%CPU utilization	%Memory utilization	Execution time (Seconds)	NTU(T) (Seconds)
6	SFMA	1	99.5	87.3	319	400
		2	98.7	89.4		
		3	98.8	88		
		4	72.4	79.9		
		5	51.3	57.4		
		6	50.6	84.1		
	Traditional	1	99.8	89.6	439	400
		2	99.7	87.4		
		3	99	73.6		
		4	99.2	88.2		
		5	51.3	57.6		
		6	98.4	87.7		