International Journal of Scientific Research & Engineering Trends
Volume 4, Issue 6, Nov-Dec-2018, ISSN (Online): 2395-566X

Review Paper of Corresponding job on Multi-core Processors

M.Tech. Scholar Anil Malviya
Dept. of Computer Science & Engg.
Patel College of Science and Technology
Indore, India
malviya.anil2@gmail.com

Assistant Professor Mr. Pritesh Jain
Dept. of Computer Science & Engg.
Patel College of Science and Technology
Indore, India
Pritesh.Arihant@Gmail.Com

Abstract **- Multi-focus processor advancement has been enhanced awesomely and it is sensibly extraordinary in execution than single focus processors in this way having the ability to engage estimation concentrated nonstop applications with correct arranging confinements. By and large standard multiprocessor nonstop arranging is stick to Sequential models which slight intra-undertaking parallelism while Parallel models, for instance, Open MP have the ability to parallelize specific areas of assignments, thusly provoking shorter response times when possible. In this paper diverse research papers have been assessed and are named Sequential Real-Time Task based Research and Parallel Real-Time Task based Research. Furthermore unique strategies, for instance, attempted part frameworks, arranging methodologies and systems used are considered for differentiating progressing errand booking in multi-focus processors.**

Keywords- **Scheduling, assignment models, errand part, undertaking arrangements, etc.**

I.INTRODUCTION

Multi-focus processors are essentially incredible in execution as appear differently in relation to single focus processors. Thusly, they can enable computation concentrated consistent applications with careful arranging confinements that can't be met on standard single-focus processors. Significantly multi-focus processors are rapidly grabbing bit of the pie with genuine chip dealers offering a routinely growing number of focuses per processor. In any case, most results in standard multiprocessor consistent arranging are limited to back to back programming models and dismissal intra-errand parallelism. From a programming perspective, the progressive programming model does not scale to a great degree well for such multi-focus systems. Parallel programming models, for instance, Open MP present promising responses for simply more effectively using distinctive processor focuses.

Genuine chip makers have starting late increment the change of colossally multi-focus processors for a grouping of reasons including power usage, memory speed confound, and rule level parallelism limits. This headway has moved the scaling designs from processor clock frequencies to the amount of focuses per processor. For example, AMD has exhibited a 12-focus Opteron [1] processor concentrating on the datacenter server publicize, while Intel has developed a 48-focus single-chip PC for appropriated figuring [2]. Intel furthermore has starting late put 80 focuses in a Teraflops Research Chip [3] with a view to making it all around available, and Clear Speed has made 96-focus processor [4]. Tilera revealed a 100-focus processor, TILE-Gx100 [5].

While hardware advancement is moving at a quick pace, programming and programming models have fail to keep pace. For example, Intel has set a period distribution of 5 years to impact their 80-to focus processor generally open due to the disappointment of current working systems and programming to manhandle the benefits of multi-focus processors [3]. As multi-focus processors continue scaling, they allow to performing more perplexing and computation genuine assignments dynamically. In any case, to take full good position of multi-focus setting up, these structures must undertaking intra-task parallelism, where parallelizable steady endeavors can utilize various focuses meanwhile. By manhandling intra-undertaking parallelism, multi-focus processors can achieve imperative steady execution change over ordinary single-focus processors for some count genuine progressing applications, for instance, video observation, radar following, and blend persistent fundamental testing [6] where the execution confinements of standard single-focus processors have been an important hindrance.

Parallel programming models, for instance, OpenMP [7], Java [8], Pthreads [9] and Cilk+ [10] are proficient plausibility for abusing future huge multi-focus processors. These models have the ability to parallelize

specific areas of endeavors, thusly provoking shorter response times when possible.

II. LITERATURE REVIEW

Diverse asks about are finished their examination in later past years for successful consistent endeavor booking. For examining counts they use diverse terms like endeavor makes, errand parameters, task needs, arranging arrangements, et cetera. Moreover extraordinary execution estimations, for instance, Utilization limits, Approximation Ratio, Resource Augmentation or Speedup factor and Empirical measures, are used to examine the reasonability of different arranging figurings.

Assignment based Review

Distinctive research papers have been investigated and are named

- Sequential Real-Time Task based Research.
- Parallel Real-Time Task based Research.

Back to back programming models ended up being extremely significant when processor producers pushed for snappier and speedier processor clock speeds. As the semiconductor shippers move the scaling designs towards more processor focuses, the upsides of progressive programming start to decrease interestingly with the weakness to abuse the available parallelism. There are distinctive Parallel programming models, e.g. OpenMP [7], are promising hopefuls that exploits without limits huge multi-focus processors. These models can parallelize particular segments of assignments.

Consecutive Real-Time Task based Research

There has been wide work on customary multiprocessor continuous arranging [11]. Most by far of this work bases on progressive programming model, on multiprocessor or multi-focus systems, where the issue is to design various sequential constant errands on various processor focuses.

S. K. Dhall and C. L. Liu [12] considered the issue of arranging incidental time-essential assignments on multiprocessor handling systems. An irregular time-fundamental endeavor contains an unbounded number of sales, each one of which has an embraced due date. The arranging issue is to decide a demand in which the requesting of a game plan of endeavors are to be executed and the processor to be used, with the goal of meeting each one of the due dates with a base number of processors. Since the issue of choosing the base number of processors is troublesome, they consider two heuristic estimations. These are definitely not hard to realize and yield different processors that is sensibly close to the base number. They moreover explore the most cynical situation lead of these heuristics.

Hard continuous structures require both for all intents and purposes alter executions and results that are conveyed on time. This infers the endeavor booking figuring is a crucial piece of these systems. K. Ramamritham, J. Stankovic, and P. Shiah [13], made beneficial booking estimations in perspective of heuristic abilities to design a course of action of assignments on a multiprocessor system. The errands are portrayed by most skeptical situation figuring times, due dates, and resources necessities. Starting with an empty deficient schedule, every movement of the request expands the present partial timetable with one of the assignments yet to be arranged. The heuristic limits used in the computation viably arrange the sweep for an achievable timetable, i.e., they help pick the errand that expands the present partial date-book.

Two booking figurings are surveyed by methods for propagation. For extending the back and forth movement mostly timetable, one of the computations considers, at every movement of the interest, each one of the endeavors that are yet to be reserved as candidates. The second focuses on a little subset of endeavors with the most concise due dates. The second computation is gave off an impression of being to a great degree convincing when the most extraordinary acceptable arranging overhead is settled. This count is in this manner fitting for dynamic getting ready for nonstop systems.

A. Khemka and R. K. Shyama sundar [14] developed a perfect booking computation and portrayed that the feasibly designs a course of action of m irregular errands on n processors before their different due dates, if the endeavor set satisfies certain conditions. The flightiness of this booking figuring similar to the amount of arranged assignments and the amount of processors and maximum points of confinement on the amount of allocations in a given time between time and for any single endeavor is in like manner decided. The perfect computation is seemed, by all accounts, to be particularly useful when designs are worked from the crucial stream regards got from the looking at most prominent stream orchestrate.

To the extent a booking redirection depiction of the issue, M. Dertouzos and A. Mok [15] discussed the issues of hard-progressing undertaking getting ready for a multiprocessor area. It is exhibited that perfect booking without from the prior data is amazing in the multiprocessor case paying little heed to whether there is no impediment on assignment inferable from need or shared dismissal objectives. Satisfactory conditions are resolved which will enable a game plan of assignments to be preferably reserved at run time.

Joseph Y.T. Leun [16] considers the versatile nature of choosing if a game plan of discontinuous, consistent assignments can be moved toward m ≥ 1 vague processor concerning settled need booking. It is exhibited that the issue is NP-hard in everything with the exception of one extraordinary case.
The diserse nature of perfect settled need arranging count is moreover discussed.

John Carpenter and Shelby Funk [17] displayed another logical order of booking figurings for arranging preemptive consistent errands on multiprocessors. They delineated some new classes of booking estimations and thought about the relationship of these classes to the present all around pondered classes. They in like manner depicted known booking computations that fall under these classes and showed sufficient credibility conditions for these figurings. In this, the trade offs related with arranging free, infrequent steady errands on a multiprocessor is in like manner separated.

III.PARALLEL REAL-TIME TASK BASED RESEARCH

There has moreover been wide work on arranging of somewhere around one parallel occupations on multiprocessors [18].
Regardless, the work in [18], [21] does not think about task due dates, and that in [22], [24] thinks about sensitive nonstop arranging. Instead of the goal (i.e. to meet all endeavor due dates) of a hard consistent system, in a sensitive progressing structure the goal is to meet a particular subset of due dates in perspective of some application specific criteria..

C. D. Polychronopoulos and D. J. Kuck [18] extensively thought about the issue of booking cycles of parallel circles among different processors in a parallel structure. They proposed the Guided self arranging strategy which watches out for the issue of uneven start times for each processor. As opposed to using a settled bump gauge, they propose lessening piece sizes, found out as a reducing limit of the present cycle number I being executed. As execution proceeds, more diminutive pieces upgrade the change of the remaining burden toward the complete of the circle.

N. S. Arora and R. D. Blumofe [19] showed a customer level string scheduler for shared-memory multiprocessors, and separated its execution under multiprogramming. They indicate multiprogramming with two booking levels: their scheduler continues running at customer level and timetables strings onto a settled amassing of strategies, while underneath this level, the working structure piece designs frames onto a settled assembling of processors. In this they see the piece as a foe, and their goal is to design strings onto systems with the ultimate objective that viable usage of whatever processor resources are given by the part ought to be conceivable. Considering the issue of arranging effectively arriving occupations in a non-smart setting, that is, the time when the degree of an employment in remains darken until the point that the action finishes execution,
N. Bansal and K. Dhamdhere [20] focused on restricting the mean log stick, where the log stick (generally called reach out) of work is portrayed as the extent of the stream time to the degree of the movement. They use resource increment with respect to empowering a speedier processor to the online count to make up for its nonappearance of data of business sizes.

Multiprocessor making arrangements for a shared multiprogramming condition can be sorted out as two-level booking, where a section level action scheduler assigns processors to occupations and a customer level string scheduler designs created by a job on the allotted processors. In this one of a kind condition, the amount of processors distributed to a particular occupation may move in the midst of the action's execution, and the string scheduler must change in accordance with these alterations in processor resources. For all things considered structure viability, the string scheduler should similarly give parallelism feedback to the movement scheduler to swear off allotting an occupation a bigger number of processors than it can use usefully.

W. J. Hsu, and C. E. Leiserson [21] gives a diagram of a couple of flexible string schedulers they have developed that give provably extraordinary history-based feedback about the action's parallelism without knowing the possible destiny of the movement. These string schedulers complete the movement in close perfect time while guaranteeing low waste. They have analyzed these string schedulers under stringent badly arranged conditions, exhibiting that the string schedulers are incredible to various system circumstances and assignment approaches. To explore the string schedulers under this not well arranged model, they have developed another framework, called trim examination, which can be used to exhibit that the string scheduler gives extraordinary lead on most by a wide margin of time steps, and performs deficiently on only a couple.

J. M. Calandrino and J. H. Anderson [22] researched diverse heuristics that undertaking to improve store execution when arranging persistent outstanding tasks at hand. Such heuristics are material when diverse multithreaded applications exist with tremendous working sets. Additionally, a relevant examination that shows how our best-performing heuristics can upgrade the end-customer execution of video encoding

applications is shown. A mutt approach for arranging continuous endeavors on broad scale multicore stages with different leveled shared stores is proposed by J. H. Anderson, and D. P. Baumberger [23]. In this methodology, a multicore organize is distributed clusters. Endeavors are statically doled out to these bundles, and arranged inside each gathering using the preemptive overall EDF booking estimation. It exhibited that this creamer of allocating and overall arranging performs favored on immense scale organizes over either approach alone. They moreover choose the best possible pack size to achieve the best execution possible, given the properties of the task set to be maintained.

J. M. Calandrino and D. Baumberger [24] discuss a methodology for supporting sensitive progressing discontinuous errands in Linux on execution lopsided multicore stages (AMPs). Such models include incalculable units on one or a couple of chips, where each getting ready unit is fit for executing a comparative course set at an other execution level. They discuss deficiencies of Linux in supporting periodic continuous assignments, particularly when focuses are unbalanced, and how such inadequacies were survived. They in like manner investigate how to give incredible execution to non-consistent assignments inside seeing a progressing outstanding burden. It is exhibited this ought to be conceivable by using deferrable servers to unequivocally hold an offer of each middle for non-consistent endeavors. This grants non-consistent endeavors to have require over progressing assignments while doing all things considered won't influence timing necessities to be manhandled, along these lines improving non-constant response times. Preliminaries exhibit that even minimal deferrable servers can significantly influence non-consistent errand execution.

There has been little work on hard consistent booking of parallel endeavors. Anderson [25] propose the possibility of a megatask as a way to deal with lessen miss rates in shared saves on multicore organizes, and consider Pfair arranging by swelling the weights of a megatask's fragment errands. Preemptive settled need booking of parallel endeavors is had all the earmarks of being NP-hard by Han in [26].
O.H. Kwon and K.- Y. Chwa [27] research preemptive EDF arranging of parallel task systems with direct speedup parallelism. In this they consider the issue of booking free parallel errands with solitary due dates with the end goal to support the total work performed by the endeavors which complete their executions beforehand due dates. They propose two polynomial-time figure computations for nonmalleable parallel errands and pliable endeavors with direct speedup.

Q. Wang and K. H. Cheng consider a heuristic for non-preemptive arranging. Regardless, this work revolves around estimations like makespan [28] or mean work that meets due date [27], and considers direct task models where an endeavor is executed on up to a given number of processors.

N. Fisher, S. Baruah, and T. P. Pastry specialist [29] showed polynomial-time estimation showed for allocating amassing of sporadic assignments among the processors of an unclear multiprocessor arrange with static-require setting up for each individual processor. Since the distributing issue is adequately seen to be NP-hard in the strong sense, this computation isn't perfect. A quantitative depiction of its most negative situation execution is given the extent that satisfactory conditions and resource increment estimation limits. The partitioning estimation is similarly surveyed over discretionarily made errand structures. Huge parts of the other work, on ceaseless arranging of parallel errands, also address distorted task models. K. Jansen [30] inspected the arranging of malleable endeavors, where every errand is relied upon to execute on a given number of focuses or processors and this number may change in the midst of execution.

W. Y. Lee and H. Lee [31] proposed an optimal(it reliably finds the conceivable logbook if one exists) computation for nonstop arranging of parallel endeavors on multiprocessors, where the assignments have the properties of versatile allocation, coordinate speedup, restricted parallelism and constrained due date. The count reliably passes on the best date-book eating up minimal processors among commonsense timetables.

G. Manimaran, C. S. R. Murthy, and K. Ramamritham [32] considered non-preemptive EDF reserving for malleable endeavors, where the genuine number of used processors is settled before starting the system and remains unaltered.Parallel programming models familiarize another estimation with this issue, where livelihoods may be part into parallel execution segments at specific core interests. Late results [33, 34] have considered particular task models for parallel programming. Sebastien Collette and Liliyana Cucu [33] investigated the overall booking of sporadic, sure due date, steady errand structures on multiprocessor stages. They gave a task show which arranges work parallelism. They exhibited that the time-multifaceted nature of the reasonableness issue of these systems is straight by and large to the amount of (sporadic) errands for a settled number of processors. They proposed a booking estimation theoretically perfect (i.e., seizures and developments overlooked). Also, they gave a right credibility utilize bound. At last, they proposed a

framework to limit the amount of developments and acquisitions.

S. Kato and Y. Ishikawa [34] address Gang EDF arranging, which applies the Earliest Deadline First (EDF) way to deal with the standard Gang booking plan, of flexible parallel errand structures. They require the customers to pick at settlement time the amount of processors whereupon a parallel errand will run. They furthermore acknowledge that a parallel errand produces unclear number of strings from processors picked before the execution. Then again, the parallel errand demonstrate tended to in this paper empowers endeavors to have assorted amounts of strings in different stages,

1.This makes our answer fitting to a significantly more broad extent of employments.

From a programming perspective, the sequential programming model does not scale to a great degree well for multi-focus systems. Parallel programming models, for instance, OpenMP show promising responses for simply more satisfactorily using distinctive processor focuses. K. Lakshmanan, S. Kato, and R. R. Rajkumar [35] ponder the issue of booking incidental consistent errands on multiprocessors under the fork-join structure used in OpenMP. They layout the speculative best-case and most cynical situation periodic fork-join errand sets from a processor utilization perspective. In light of impression of these errand sets, a divided settled need arranging count for discontinuous fork-join endeavors is given. The proposed multiprocessor booking count is seemed to have a benefit increment bound of 3:42, which induces that any errand set that is achievable on m unit speed processors can be arranged by the proposed estimation on m processors that are 3:42 times speedier.

For depicting the execution examination of booking computations, usually utilize limits, for instance, those used by C. L. Liu and J.W. Layland [37] are used. Regardless, there exist errand sets with a total utilize to some degree more unmistakable than and discretionarily close to 1 that are unschedulable on a system with m processor focuses, for this standard utilize points of confinement may not be important for analyzing the execution of the arranging figurings for such endeavor sets. Resource extension limits, for instance, those showed S. Funk, J. Goossens, and S. Baruah give off an impression of being prepared contender for the execution examination of booking figurings.

The creating hugeness of parallel endeavor models for persistent applications introduces new challenges to consistent booking speculation that has by and large revolved around progressive errand models. Prominently, K. Lakshmanan [35] in his work on parallel anticipating progressing endeavors looks at the benefit development bound using allotted Deadline Monotonic (DM) booking, and does not think about other arranging courses of action, for instance, overall EDF.

Besides, K. Lakshmanan [35], considers a synchronous endeavor display (a key Fork-Join illustrate), where each parallel errand involves a movement of back to back or parallel parts. This model can be called synchronous, since each one of the strings of a parallel part should finish before the accompanying segment starts, making a synchronization point. In any case, that endeavor show is restrictive in that, for every task, each one of the pieces have a proportional number of parallel strings, and that number must not be more imperative than the total number of processor focuses. While the work acquainted talks with a promising development towards parallel steady setting up for multi-focus processors, the imprisonments on the endeavor show make the courses of action prohibited for some progressing applications that much of the time use unmistakable amounts of strings in different bits of count.

Likewise, an endeavor stretch or errand crumbling computation is used which separates each parallel task into a course of action of progressive endeavors, that impacts a pro to string of execution essential equal to errand period, and designate one processor focus exclusively to it. Whatever remains of the strings are reserved using FBB-FDD estimation. Their results don't hold if, in an endeavor, the amount of strings in different pieces move, or outperform the amount of focuses. Consequently may not be direct material to more expansive task models.

These obstacles are overpowered by considering a more summed up synchronous errand show by Abusayeed Saifullah [36] instead of the restrictive endeavor display tended to in [35], where a general synchronous parallel task exhibit considered where every errand includes sections, each having a self-self-assured number of parallel strings. In like manner a novel endeavor rot count is prescribed that crumbles each parallel errand into a course of action of successive assignments. Each bit may contain an optional number of parallel strings. That is, unmistakable segments of a comparable parallel errand can contain assorted amounts of strings, and pieces can contain a greater number of strings than the amount of processor focuses. This model is more helpful, since a comparable endeavor can be executed on machines with pretty much nothing and likewise significant amounts of focuses.

In future, a general fork-join show with additionally created segment, for instance, settled fork-join structures

can in like manner be made. Also, an endeavor disintegration figuring that may be material to more wholes up errand model can similarly be made. Besides, the benefit development bound considering other booking approaches, for instance, overall EDF can be examined.

Relative Study of various steady multiprocessor estimations various progressing figurings are considered and are contemplated dependent on errand or models used. Table 1 and Table 2 exhibit the examination between various figurings.

Table 1 Comparative study of various real time multiprocessor algorithms meant for Sequential Task/Model.

S. N.	Algo. Name	Task Splitting Tech. / Sche. Policy/ Used Technique
1.	RM-US	Categories tasks as heavy light based on certain threshold
2.	SM-US	Categories tasks as heavy light and priority order is given by SM
3.	HSA	Uses heuristic function to search tasks to be scheduled
4.	MA	It focuses attention on a small subset of tasks with the shortest deadlines set of m periodic tasks on n
5.	SAN	processors before their respective
6.	OFPSA	Discusses the problem of scheduling periodic real time tasks

Table 2 Comparative study of various real time multiprocessor algorithms meant for Parallel Task/Model.

S.N.	Algo. name	Task Splitting Tech. / Sche. Policy/ Used Technique
1.	GSS	It uses decreasing chunk sizes, where smaller chunks improve the balance of workload towards end of loop.
2.	ULTS	Modeled multiprogramming with two levels at user level on to a fixed collection of processes and at kernel level on to a fixed collection of processors.
3.	NCS	Use resource augmentation in terms of allowing a faster processor to the online algorithm to make up for its lack of knowledge of job sizes.
4.	ASPF	Based on trim analysis technique.
5.	HRTS-LSM	Uses hybrid approach for scheduling with hierarchical shared caches.
6.	PRTTS	Uses concept of mega task.
7.	H-Pfair	It finds an approximate job partition on two processors.
8.	FBB-FFD	Provides sufficient conditions for feasibility, a resource augmentation approximation ratio, and simulation results.
9.	EDF-ID	Schedules independent parallel tasks with individual deadlines.
10.	Opt-Algo	Finds feasible schedule using fewest processors.
11.	NP-EDF	Number of used processors is defined before starting the system.
12.	GSA-ST	Uses job parallelism on identical parallel machines.
13.	Gang-EDF	Derives schedulability test on the basis of Global EDF schedulability test i.e. BAR test.
14.	FPP-FJS	Developed synchronous task stretch model using Fork-Join task sets.
15.	DP-FJS	Developed general parallel synchronous task stretch model using Fork-Join task sets

IV. CONCLUSION

In this audit paper, different research papers have considered based on different methodologies, for example, errand part strategies, planning arrangements and procedures utilized for contrasting continuous undertaking planning for multi-center processors and are classified as Sequential Real-Time Task based Research and Parallel Real-Time Task based Research. Regardless of whether broad work done as of now around there, a gigantic degree for research on parallel models is still there. This audit of existing calculations is to uncover the examination challenges existing in this field of ongoing assignment booking particularly on multi-center processors.

REFERENCES

[1]. "AMD sets the new standard for price, performance, and power for the datacenter," AMD Press Release, March 2010.
[2]. "Single-chip cloud computer," Intel Research, Dec 2009.
[3]. "Teraflops research chip," http://techresearch .intel. com/ProjectDetails.aspx ?Id = 151.
[4]. "CoSy compiler for 96-core multi-threaded array processor,"http://www.clearspeed.com /newsevents/ news/ClearSpeedAce 011708.php.

[5]. "Comming soon tile-gx100 the first 100 cores processors in the world," http:// internalcomputer. com/coming-soon-tilegx100-the-first-100-cores-rocessor-in-the-world. computer , Feb 2011.
[6]. H.-M. Huang, T. Tidwell, C. Gill, C. Lu, X. Gao, and S. Dyke, "Cyber-physical systems for real-time hybrid structural testing: a case study," in ICCPS '10.
[7]. OpenMP," http://openmp.org.
[8]. D. Lea, "A java fork/join framework," in Proceedings of the ACM 2000 Java Grande Conference, p. 3643, June 2000.
[9]. Adrien Lamothe. Pthreads programming: A hands-on introduction, 2007.
[10]. "IntelR CilkTMPlus," http:// software. intel. com/ en-us/articles/intel-cilk-plus
[11]. R. Davis and A. Burns, "A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems," University of York, Department of Computer Science, Tech. Rep. YCS-2009-443, 2009.
[12]. S. K. Dhall and C. L. Liu, "On a real-time scheduling problem," OPERATIONS RESEARCH, vol. 26, 1978.
[13]. K. Ramamritham, J. Stankovic, and P. Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems," IEEE Transactions on Parallel and Distributed Systems, vol. 1, pp. 184–194, 1990.
[14]. A. Khemka and R. K. Shyamasundar, "An optimal multiprocessor real-time scheduling algorithm," J. Parallel Distrib. Comput., vol. 43, no. 1, pp. 37–45, 1997
[15]. M. Dertouzos and A. Mok, "Multiprocessor online scheduling of hard-real-time tasks," IEEE Transactions on Software Engineering, vol. 15, pp. 1497–1506, 1989.
[16]. J. Y.-T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," Performance Evaluation 2, p. 237250, Dec 1982.
[17]. J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, "A categorization of real-time multiprocessor scheduling problems and algorithms," in Handbook on Scheduling Algorithms, Methods, and Models. Chapman Hall/CRC, Boca, 2004.
[18]. C. D. Polychronopoulos and D. J. Kuck, "Guided selfscheduling: A practical scheduling scheme for parallel supercomputers," IEEE Transactions on Computers, vol. C-36, no. 12, pp. 1425–1439, 1987.
[19]. N. S. Arora, R. D. Blumofe, and C. G. Plaxton, "Thread scheduling for multiprogrammed multiprocessors," in SPAA '98.
[20]. N. Bansal, K. Dhamdhere, J. Konemann, and A. Sinha, "Nonclairvoyant scheduling for minimizing mean slowdown," Algorithmica, vol. 40, no. 4, pp. 305–318, 2004.
[21]. K. Agrawal, Y. He, W. J. Hsu, and C. E. Leiserson, "Adaptive task scheduling with parallelism feedback," in PPoPP '06.
[22]. J. M. Calandrino and J. H. Anderson, "Cache-aware realtime scheduling on multicore platforms: Heuristics and a case study," in ECRTS '08.
[23]. J. M. Calandrino, J. H. Anderson, and D. P. Baumberger, "A hybrid real-time scheduling approach for large-scale multicore platforms," in ECRTS '07.
[24]. J. M. Calandrino, D. Baumberger, T. Li, S. Hahn, and J. H. Anderson, "Soft real-time scheduling on performance asymmetric multicore platforms," in RTAS '07.
[25]. J. H. Anderson and J. M. Calandrino, "Parallel real-time task scheduling on multicore platforms," in RTSS '06.
[26]. C.-C. Han and K.-J. Lin, "Scheduling parallelizable jobs on multiprocessors," in RTSS '89.
[27]. O.H. Kwon and K.-Y. Chwa, "Scheduling parallel tasks with individual deadlines," Theor. Comput. Sci., vol. 215, no. 1-2, pp. 209–223, 1999.
[28]. Q. Wang and K. H. Cheng, "A heuristic of scheduling parallel tasks and its analysis," SIAM J. Comput., vol. 21, no. 2, pp. 281–294, 1992.
[29]. N. Fisher, S. Baruah, and T. P. Baker, "The partitioned scheduling of sporadic tasks according to static-priorities," in ECRTS. IEEE Computer Society, 2006, pp. 118–127.
[30]. K. Jansen, "Scheduling malleable parallel tasks: An asymptotic fully polynomial time approximation scheme," Algorithmica, vol. 39, no. 1, pp. 59–81, 2004.
[31]. W. Y. Lee and H. Lee, "Optimal scheduling for real-time parallel tasks," IEICE Trans. Inf. Syst., vol. E89-D, no. 6, pp. 1962–1966, 2006.
[32] G. Manimaran, C. S. R. Murthy, and K. Ramamritham, "A new approach for scheduling of parallelizable tasks in realtime multiprocessor systems," Real-Time Syst., vol. 15, no. 1, pp. 39–60, 1998.
[33]. S. Collette, L. Cucu, and J. Goossens, "Integrating job parallelism in real-time scheduling theory," Inf. Process. Lett., vol. 106, no. 5, pp. 180–187, 2008.
[34]. S. Kato and Y. Ishikawa, "Gang EDF scheduling of parallel task systems," in RTSS '09.
[35]. K. Lakshmanan, S. Kato and R. R. Rajkumar, "Scheduling parallel real-time tasks on multi-core processors," in RTSS '10.
[36]. Abusayeed Saifullah, Kunal Agrawal, Chenyang Lu, and Christopher Gill, "Multi-core Real-Time

Scheduling for Generalized Parallel Task Models", in RTSS'11.

[37]. C. L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," J. ACM, vol. 20, no. 1, pp. 46–61, 1973.

[38]. S. Funk, J. Goossens, and S. Baruah, "On-line scheduling on uniform multiprocessors," in Proceedings of the IEEE Real-Time Systems Symposium (December 2001), IEEE Computer. Society Press, 2001, pp. 183–192.