Una Clasificación General de las Estrategias de Distribución de Carga

Javier Echaiz* Jorge R. Ardenghi

Laboratorio de Investigación en Sistemas Distribuidos (LISiDi)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur – Bahía Blanca, Argentina
T.E.: +54 291–4595135 Fax: +54 291–4595136
e-mail: {je, jra}@cs.uns.edu.ar

Resumen

Recientemente la computación distribuida sobre estaciones de trabajo se ha convertido en una alternativa barata y efectiva para hacer frente a la performance ofrecida por las supercomputadoras. Estos sistemas construidos con hardware estándar de producción masiva se encuentran ampliamente disponibles tanto en ambientes académicos como industriales, sin embargo para lograr explotar al máximo su potencial se vuelve necesaria la presencia de un algoritmo de distribución de carga efectivo.
Diferentes tipos de aplicaciones y entornos de cómputo requieren diferentes algoritmos de distribución de carga, es entonces indispensable contar con una clasificación que provea la terminología necesaria y una metodología de comparación eficaz y útil para comparar y elegir la mejor estrategia aplicable al entorno subyacente.
En este trabajo se propone una nueva y más general clasificación de las estrategias de distribución de carga. Además se presentan ejemplos que verifican su funcionamiento y utilidad para la creación de nuevos enfoques.

Términos Clave: computación distribuida, carga compartida, balance de carga, estrategias.

1 Introducción

Carga compartida o balance de carga es una estrategia empleada para distribuir carga entre nodos pertenecientes a un sistema distribuido. Algunos autores ([ELZ86, KL88, OJ92]) hacen la distinción entre estas dos. El *balance de carga* se define como una estrategia que continuamente trata de igualar (dentro de un determinado margen) la carga en cada nodo del sistema distribuido. Por otro lado, los que soportan *carga compartida*, tratan de repartir la carga del sistema tratando de evitar la sobrecarga de nodos individuales. Para simplificar emplearemos el término distribución de carga para referirnos tanto a carga compartida como a balance de carga. Este término más general fue introducido por primera vez por Jacqmot y Milgrom [JM93].

El objetivo principal de la distribución de carga es lograr un uso más eficiente de los recursos (usualmente el CPU) tratando de asegurar que ningún nodo esté ocioso. Esta tarea se logra transfiriendo carga (procesos) desde un nodo ocupado hacia un nodo ocioso/ligeramente cargado.

Además de buscar incrementar la performance global, un sistema de distribución de carga tiene como meta minimizar el costo de la comunicación entre nodos que se origina al acceder

*Becario de la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, Argentina.

a recursos remotos. Otros objetivos incluyen mejorar la performance suavizando los picos de sobrecarga y lograr aliviar problemas en sistemas de tiempo real, donde algunas tareas no pueden cumplir con su deadline aún en los casos en los que el sistema tiene la capacidad de cumplir con todos los deadlines. Esto último puede ocurrir cuando la llegada de nuevas tareas no está uniformemente distribuida, causando que algunos nodos estén sobrecargados mientras que al mismo tiempo otros están ociosos. Existen muchas otras razones presentes en la literatura, sin embargo independiente del por qué o cómo se lleva a cabo la distribución de la carga, debe preferentemente realizarse en forma transparente de forma tal de ocultar al usuario la distribución y multiplicidad de los recursos [EA03].

Las estrategias de distribución de carga constituyen indudablemente el punto más crítico en la implementación eficiente de diversos sistemas de computación distribuidos. Cuando la utilización de los nodos del sistema es dispar, el desbalance de la carga constituye un causante potencial de problemas de performance.

Existen diversos factores que dependiendo de la aplicación y del entorno considerado pueden impactar negativamente el balance de la carga. Dentro de los llamados factores estáticos podemos señalar a modo de ejemplo la heterogeneidad de los procesadores. En el grupo de los factores dinámicos se encuentran aspectos como el desconocimiento del costo computacional de cada tarea, la creación y migración de tareas, y la variación dinámica de los recursos computacionales debido a cargas externas.

Durante las dos últimas décadas se han propuesto tantas y tan diversas estrategias de distribución de carga que no es tarea fácil para el diseñador comparar y seleccionar la estrategia más efectiva que se adecúe a un determinado entorno. Los algoritmos van desde los destinados a sistemas multiprocesador fuertemente acoplados hasta los que trabajan sobre sistemas distribuidos para WANs. Dada esta multiplicidad, un método general para clasificar la diversidad de diseños es extremadamente valuable. Una clasificación efectiva debe ser capaz de:

1. Resaltar las características esenciales de un sistema de distribución de carga para posibilitar el análisis y la comparación de las diferentes propuestas.
2. Proveer de terminología consistente y organizar el conocimiento para tratar de identificar oportunidades para nuevos enfoques.

En este trabajo se pretende presentar una nueva clasificación que cumpla con estos dos objetivos. Clasificaciones previas no sólo persiguen diferentes metas sino que además apuntan a ciertas aplicaciones particulares, a diferencia de nuestra propuesta que trata de ser general. Por ejemplo [CK88], se centra en el *scheduling* de procesos en sistemas operativos distribuidos y en el scheduling de tareas en aplicaciones paralelas, basándose en una descomposición funcional. Por otra parte [PRR01] clasifica estrategias para distribución de carga en aplicaciones SPMD.

La clasificación aquí propuesta incorpora algoritmos de distribución de carga de diferentes características, estrategias diseñadas para sistemas homogéneos y heterogéneos, migración de tareas, algoritmos centralizados y distribuidos, etcétera.

La siguiente sección compendia brevemente clasificaciones previas. En la Sección 3 se describe la clasificación de políticas propuesta. A continuación, en la Sección 4, se presentan algunos algoritmos de distribución de carga conocidos y su caracterización dentro de nuestra clasificación. Por último, la Sección 5 resume las conclusiones obtenidas y plantea algunas posibilidades de trabajos futuros.

2 Clasificaciones Previas de Distribución de Carga

La diversidad de sistemas distribuidos es probablemente atribuible a la necesidad de adaptación a las diferentes arquitecturas y ambientes. Consecuentemente muchos trabajos han tratado, al menos hasta cierto punto, de clasificar sistemas previos y de introducir terminología para poder permitir comparaciones constructivas. Esto trajo aparejado confusión como resultado debido a la imposibilidad de hacer comparaciones sencillas, producto de una marcada falta de rigurosidad y de terminología estándar.

Esta sección describe brevemente las aproximaciones significativas a la clasificación y entendimiento de los algoritmos de distribución de carga, comenzando con la notable taxonomía presentada en [CK88]. Las clasificaciones anteriores a la recién mencionada tienden a ser incompletas y por esta razón limitaremos nuestra discusión a trabajos posteriores[1].

2.1 Taxonomía de Casavant y Kuhl

Casavant y Kuhl [CK88] presentaron un trabajo pivotal acerca de una clasificación de distribución de carga basada en decisiones de diseño, tales como estático vs. no estático y distribuido vs. no distribuido. Su taxonomía combina estas decisiones en una clasificación jerárquica aumentada con una clasificación horizontal. La división entre las clasificaciones jerárquica y horizontal responde a características que no caen únicamente en una rama determinada de la jerarquía pero que no obstante son importantes para describir el comportamiento del sistema de distribución de carga.

Sin embargo, las clases que constituyen la clasificación horizontal tales como *bidding* y la elección entre migración de procesos no apropiativa y apropiativa son inherentemente distintas de aquellas de la jerarquía. La mayor parte de estas características definidas por la clasificación horizontal son técnicas en lugar de ser elecciones de políticas de diseño.

Esta clasificación fue diseñada para poder ubicar un sistema de distribución de carga en una clase bien definida. Los autores probaron la efectividad de su taxonomía clasificando varios algoritmos conocidos. Sin embargo, esta clasificación no puede utilizarse como herramienta de comparación en varios casos de algoritmos pertenecientes a diferentes sistemas de distribución de carga.

En [BW90] se identifica un problema en la taxonomía de Casavant y Kuhl. En este trabajo se considera que las clasificaciones de estrategias y problemas se encuentran entremezcladas en lugar de estar claramente separadas. En este trabajo los autores tomaron la taxonomía original y separaron la clasificación de problemas en una nueva clasificación, denominada *ESR* (*Event* (Evento), *Surroundings* (Entorno) y *Requirements* (Requerimientos)). El resto de la clasificación original constituyó la clasificación resultante.

La clasificación ESR incluye un número de mecanismos dentro de la clasificación de problemas. Por ejemplo los autores sugieren que la elección entre migración apropiativa y no apropiativa pertenece a la clase S por ser una capacidad del entorno. Si bien todavía no es clara la diferencia de performance entre un sistema que implementa un estilo de migración y el otro, es un item que indudablemente debe ser considerado de alguna manera en la caracterización de un algoritmo de distribución de carga.

[1] En las secciones bibliográficas de los trabajos aquí citados pueden encontrarse referencias a trabajos anteriores.

2.2 Clasificaciones Funcionales

La contrastante clasificación presentada por Jacqmot y Milgrom [JM93] es más especializada que la propuesta en [CK88]. En esta clasificación se consideran dos características ortogonales presentes en muchos sistemas de distribución de carga: movimiento de procesos (PM) y manejo de la información (IM). Para cada una de estas estrategias se aislaron unidades funcionales básicas, como la identificación del nodo fuente (bajo la estrategia PM), que se combinan para formar clases que pueden describir un sistema de forma más precisa. Estas familias se construyen basándose en el orden en el que se aplican en el sistema las unidades funcionales. Por ejemplo, bajo la estrategia PM, si el proceso candidato (y por lo tanto implícitamente el nodo fuente) es identificado antes que el nodo destino, entonces pertenece a una familia PM diferente de la que selecciona el nodo destino antes de identificar los procesos candidatos (implícitamente el nodo fuente).

En [WM85] también fueron consideradas las mismas dos categorías mayores, i.e., el orden de las acciones y la información requerida para asegurar el cumplimiento de las políticas de distribución de carga. Sin embargo en este trabajo no se presenta una descomposición en unidades funcionales.

Un problema que surge en ambas clasificaciones funcionales se produce cuando se consideran sistemas *simétricamente iniciados* como el propuesto en [BMD94]. En este caso la distribución de la carga es iniciada tanto por el emisor como por el receptor y por lo tanto no puede ajustarse a las familias presentadas en estas clasificaciones. Por otra parte, la mayoría de los sistemas pertenecen a la clase de los *iniciados por el emisor*, e invariablemente la identificación del nodo fuente precede a la del nodo destino. Como consecuencia, y resaltando su mayor inconveniente, estas clasificaciones proveen muy poca diferenciación entre los miembros de estas dos grandes clases.

2.3 Clasificación por Composición

En [BH99] se presenta una taxonomía funcional para sistemas de distribución de carga basada en dos grandes clases: *políticas* y *mecanismos*. Las políticas constituyen el conjunto de las decisiones de diseño necesarias para cumplir los objetivos del sistema. Los mecanismos implementan la distribución de la carga y proveen toda la información requerida por las políticas.

En este trabajo se identifican tres tipos de decisiones (políticas) que deben tomarse: participación, selección de ubicación, y selección de candidato. Estas decisiones determinan básicamente qué nodos participan y qué nodos y procesos candidatos están involucrados en la distribución de la carga. Existen variados mecanismos capaces de soportar estas políticas. En estos últimos años la atención de investigadores del área fue captada por temas como transparencia, dependencias residuales, performance y complejidad, aspectos que pueden reconocerse como mecanismos capaces de soportar uno de tres roles clave relacionados con la carga: transferencia, métrica, y comunicación de la métrica.

Si bien esta clasificación logra su objetivo principal, i.e., separar claramente las políticas de los mecanismos, no es suficientemente específica a la hora de caracterizar en forma precisa todas las características de un sistema de carga distribuida. Por ejemplo, no distingue dónde toman lugar las decisiones (e.g. localmente o globalmente), o si el algoritmo de distribución es sincrónico o asincrónico.

2.4 Otras Clasificaciones

En esta sección se delineó el estado actual en la clasificación de sistemas de distribución de carga. Por cuestiones de espacio no es posible presentar un informe completo acerca de todos los trabajos en este campo. Sin embargo hay varias direcciones que merecen atención. En [JV88] se introduce una clasificación basada exclusivamente en características, tales como transparencia, costo y apropiación. Contrario a este enfoque, en [LMR91] se describe una taxonomía basada en el alcance de las decisiones y migraciones, i.e., si una decisión o migración involucra solamente a los nodos directamente conectados o a todos los nodos del sistema.

Es evidente que las clasificaciones previas distan de una u otra forma en el cumplimiento de los objetivos planteados en la Sección 1. La siguiente sección presenta de forma detallada nuestra propuesta.

3 Clasificación General de Estrategias

Esta nueva clasificación intenta definir un algoritmo de distribución de carga en forma completa. Para ello presenta cinco ramas principales:

Invocación: Determina cuándo comienza a ejecutarse el mecanismo involucrado en la distribución de la carga.

Ubicación: Especifica dónde se ejecuta el algoritmo de distribución de carga.

Toma de Decisiones: Determina desde dónde (nodo(s)) proviene la información que se empleará para la toma de decisiones.

Políticas de Comunicación: Establece la adyacencia y la topología de los nodos, y determina la forma de intercambiar la carga entre ellos. Además mueve físicamente la carga entre el nodo fuente y el nodo destino.

Selección de la Carga: Decide qué nodos y procesos serán seleccionados para el intercambio de la carga.

En la Figura 1 se muestra un esquema de estas grandes ramas y sus interrelaciones. En el resto de la sección se describe detalladamente cada una de ellas. La Figura 2 esquematiza la clasificación propuesta[2].

3.1 Invocación

La estrategia de invocación especifica **cuándo** comienza el mecanismo involucrado en las actividades de carga distribuida. A su vez, la invocación puede ser *periódica*, o *basada en eventos*. La invocación periódica está basada en un *timer*, i.e., la información de carga se intercambia cada vez que transcurre un predeterminado período de tiempo. Por el contrario, los algoritmos basados en eventos usualmente emplean una estrategia dependiente de la carga observada localmente.

Las estrategias dependientes de la carga pueden ser *iniciadas por el emisor* [Rub87, DO91], *iniciadas por el receptor* [LM82, LO86] o un híbrido entre ambas denominado *iniciadas simétricamente* [ELZ86, BMD94]. En las iniciadas por el emisor los nodos sobrecargados inician los

[2]Por cuestiones de espacio la figura no muestra en la rama Políticas de Comunicación una alineación horizontal correcta entre nodos de igual nivel.

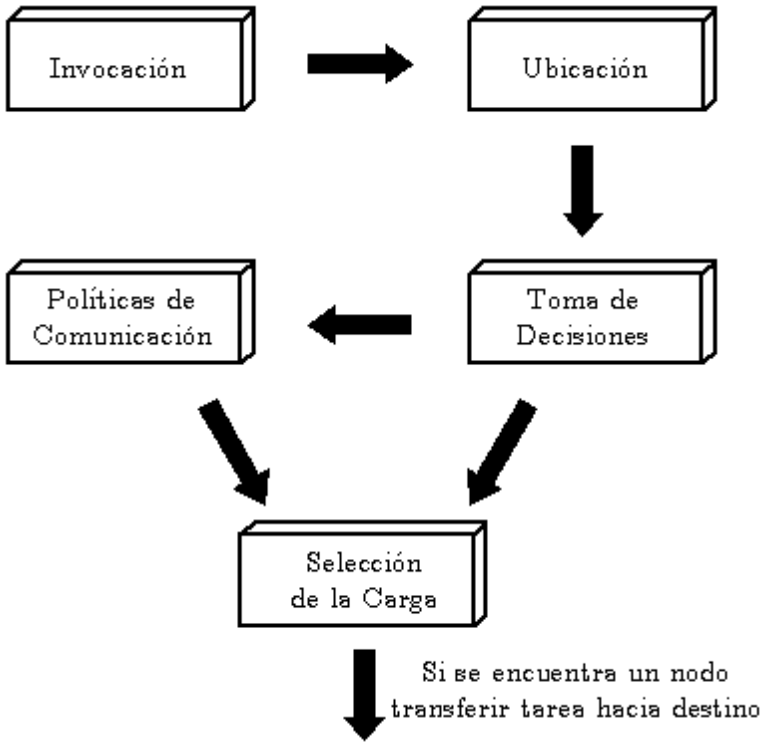

Figura 1: Relaciones entre los Componentes de la Clasificación Propuesta.

mecanismos de distribución, mientras que son los nodos con carga liviana o moderada quienes lo hacen en estrategias iniciadas por el receptor. Estos métodos de invocación pueden combinarse para crear la tercer subcategoría, tratando así de lograr una mejor reacción a los cambios de carga. Eager et. al. [ELZ86] compararon las políticas de iniciación iniciadas por el emisor y por el receptor y concluyeron que las primeras son superiores en sistemas con cargas ligeras a medias y que las iniciadas por el receptor obtienen mejores resultados en sistemas sobrecargados.

Condiciones de estado del nodo del tipo ligeramente cargado o sobrecargado son definidas generalmente por medio de dos *umbrales*: un umbral L define el punto en el que un nodo empieza a descargar el exceso de carga y un umbral H que marca el punto donde no se acepta más carga remota. Si la carga de un nodo se vuelve menor que L entonces se dice que el nodo está ligeramente cargado; si la carga está entre L y H entonces presenta una carga normal y si sobrepasa H entonces se trata de un nodo sobrecargado.

Las estrategias basadas en eventos responden mejor a los desbalances de carga, mientras que las estrategias periódicas son más sencillas de implementar. Sin embargo, las periódicas pueden traer aparejados overheads adicionales cuando las cargas están balanceadas y por lo tanto la distribución de carga no sólo no sería necesaria sino que en este caso indudablemente impactaría en el sistema en forma perjudicial.

3.2 Ubicación

La estrategia de ubicación determina **dónde** se ejecuta el algoritmo de distribución de carga. En este sentido se puede clasificar un algoritmo en *centralizado*, *distribuido*, o *híbrido*. En el caso del centralizado el algoritmo se ejecuta en un único nodo, determinando las transferencias de carga necesarias e informando a los nodos involucrados. Por otra parte, cuando el algoritmo es distribuido todos los nodos toman parte en las decisiones de distribución. Obviamente, en

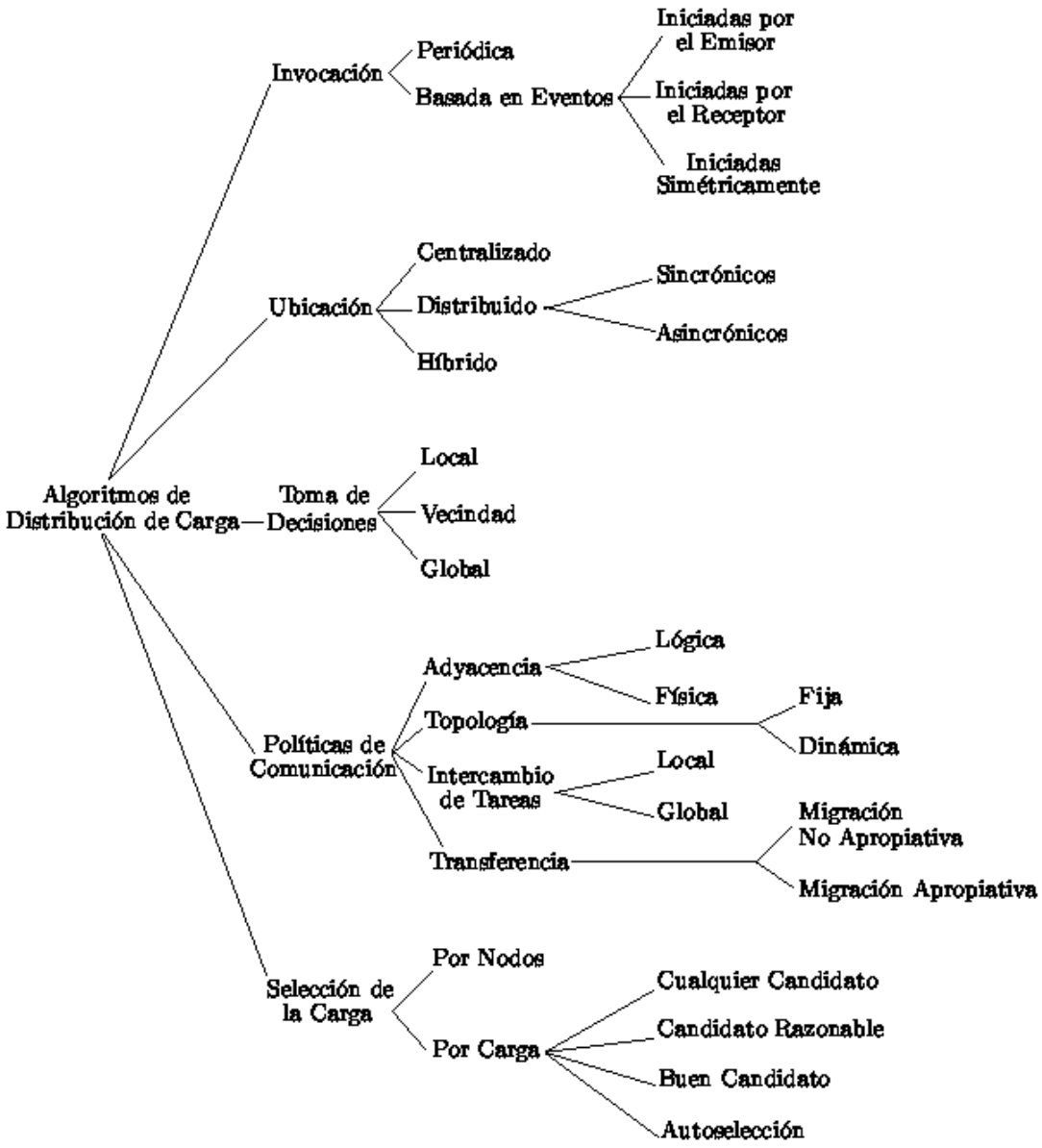

Figura 2: Clasificación Propuesta.

el caso de los híbridos se trata de lograr un punto intermedio entre los dos métodos anteriores que puede ser provechoso en ciertos ambientes.

A su vez los algoritmos distribuidos se subdividen en *sincrónicos* y *asincrónicos*. Un algoritmo sincrónico debe ejecutarse simultáneamente en todos los nodos participantes del sistema. Cuando un algoritmo sincrónico es invocado, los nodos detienen el procesamiento de sus aplicaciones para dar comienzo a los mecanismos de distribución. Generalmente este tipo de algoritmo se utiliza en entornos donde el procesamiento de las aplicaciones debe obligatoriamente detenerse en puntos de sincronización. En cambio, en los asincrónicos, el algoritmo puede ejecutarse en cualquier momento en un determinado nodo, independientemente de lo que se está ejecutando en el resto de los nodos del sistema.

Si bien el empleo de un recurso centralizado, en este caso un procesador de un nodo, puede potencialmente crear cuellos de botella, es importante recordar que las estrategias distribuidas requieren que la información de carga se propague a todos los nodos, tendiendo en general a

costos mayores de comunicación.

3.3 Toma de Decisiones

En esta estrategia se determina **de dónde** se obtiene la información utilizada por el algoritmo de distribución de carga. Las posibilidades en este punto son tres: *local*, *vecindad* (o región), y *global*. Local implica que la única información considerada es la presente en el propio nodo, y es aplicable a algoritmos donde no importa (no se considera) la situación del resto de los nodos. Por ejemplo para un dado algoritmo un nodo que se encuentra sobrecargado puede decidir disminuir su carga enviando un proceso hacia otro nodo al azar. En las estrategias de vecindad el nodo recibe información de nodos adyacentes. Si en cambio, la información se obtiene globalmente el sistema se encarga de "recolectarla" de todos los nodos.

Aunque las estrategias que consideran la información en forma local tienden a disminuir el costo de comunicación, las globales suelen propiciar decisiones más precisas. Las técnicas de vecindad tratan de obtener lo mejor de ambos mundos.

3.4 Políticas de Comunicación

La política de comunicación establece **cómo** están dispuestos los nodos dentro del sistema, i.e., define la *adyacencia* y la *topología*. La adyacencia (esencial para algoritmos cuyos nodos se comunican con su vecindad) puede a su vez ser *lógica*, e.g. estructura jerárquica de nodos donde no necesariamente todos los nodos se encuentran en contacto directo, o *física*, donde invariablemente la adyacencia es real, i.e., los nodos son parte de la misma LAN. En cambio, la topología determina el dinamismo de los vecinos de cada nodo y cabe aclarar que es independiente de la adyacencia previamente mencionada. Una topología *fija* indica un conjunto fijo de nodos vecinos, mientras que en una topología *dinámica* el nodo elige dinámicamente (en tiempo de ejecución) otro nodo (o conjunto de nodos) con quien intercambiar información.

Además la política de comunicación es responsable de especificar el *intercambio de tareas/carga* entre los diferentes nodos. Si se emplea una estrategia *global*, las transferencias de tareas/carga pueden darse entre cualquier par de nodos del sistema, mientras que una estrategia *local* define grupos de nodos, restringiendo las transferencias a pares de nodos del mismo grupo.

Relacionado en este caso con los procesos en lugar de con los nodos, la política de *transferencia* es responsable de mover físicamente las tareas/carga desde el nodo fuente hasta el nodo destino. La distribución de la carga puede ocurrir en una de dos fases: las tareas pueden asignarse a un nodo antes de empezar su ejecución (*migración no apropiativa*, también llamada ejecución remota) o pueden ser transferidas en tiempo de ejecución (*migración apropiativa*). Estas dos formas de migración constituyen para algunos investigadores las piedras basales de un esquema de clasificación. Si bien es claro que constituyen una parte importante en la política de comunicación, creemos que no debe restarse importancia al resto de las estrategias de comunicación. Más aún, debemos recordar que la elección entre un estilo de migración y el otro no es una tarea sencilla, pues depende fundamentalmente de los objetivos perseguidos por el sistema, el entorno de ejecución y los costos que pueden afrontarse en cuanto a dificultad de implementación.

3.5 Selección de la Carga

La política de selección de la carga determina **quiénes** (nodos y procesos) serán seleccionados. En primer término se determinan los nodos involucrados en el intercambio de la carga (política *por nodos*). Por ejemplo si se emplea una invocación iniciada por el emisor sólo se debe elegir el nodo destino, inversamente si la invocación es iniciada por el receptor sólo resta encontrar (seleccionar) un nodo fuente.

Además esta política especifica los items de carga apropiados para el intercambio (*por carga*). Esta rama presenta cuatro posibles opciones para seleccionar los procesos candidatos:

Cualquier Candidato: se elige un proceso al azar para migrarlo apropiativamente o se selecciona el próximo proceso para ejecutarlo remotamente. Los procesos seleccionados de esta forma no siempre son útiles para la migración, e.g. aquellos que no se ejecutan durante un período de tiempo suficiente como para justificar el overhead que implica transferir un proceso de un nodo a otro.

Candidato Razonable: se eliminan aquellos procesos que son malos candidatos para la distribución de carga. El procedimiento habitual para esta tarea se basa en el tiempo de ejecución promedio de cada proceso.

Buen Candidato: es un criterio más rígido y más costoso, pues se busca alguna forma de concordancia entre los recursos disponibles en el nodo destino y los requerimientos de recursos del proceso candidato. Más aún, en algunos sistemas esta selección del candidato cobra mayor importancia que la política de selección de los nodos, e.g. en Emerald [Leh93].

Autoselección: se basa en el concepto de que el propio proceso es quien tiene mayor capacidad para reconocer sus necesidades de recursos y que por lo tanto se encuentra en una posición ideal para hacer la mejor elección. Además del candidato, el sistema aún debe considerar el estado de los nodos destino potenciales.

La clasificación arriba descripta está más orientada a colaborar en la comparación de algoritmos de distribución de carga que en producir la clasificación específica de un sistema. Cualquier sistema involucrará (de forma explícita o implícita) a la mayoría de las hojas del árbol planteado en la Figura 2. Las decisiones de diseño no siempre serán independientes, pues por ejemplo no podrá emplearse un algoritmo de Selección de Carga "Buen Candidato" si se tiene Toma de Decisiones Local.

Cada componente de un sistema de distribución de carga representado en la clasificación puede ser dependiente del sistema subyacente, o puede ser simplemente una elección personal del diseñador o más aún, puede estar basado en la facilidad de implementación. Sin embargo creemos que de todas formas la diversidad de aproximaciones, restricciones y soluciones vuelve a esta clasificación una herramienta interesante.

4 Clasificación de Algunos Algoritmos de Distribución de Carga

En esta sección mencionaremos brevemente el comportamiento de algunos algoritmos de distribución de carga conocidos e ilustraremos de qué forma cada algoritmo encaja en la clasificación propuesta.

Algoritmo Centralizado [Hil85]

En primer término este algoritmo computa la carga promedio y luego se la envía mediante broadcast al resto de los nodos del sistema. Luego clasifica a los nodos en una de tres clases: ocioso, sobrecargado, y otros. El objetivo del algoritmo es tratar de equiparar (balancear) la carga de cada nodo sobrecargado con un par ocioso.

Algoritmo Aleatorio [SS94]

Cada vez que se crea un item (tarea) en un nodo, éste es enviado a otro nodo en forma aleatoria. La probabilidad de recibir tarea adicional en cada nodo es la misma sin importar su ubicación en el sistema.

Algoritmo del Vecindario Directo (VD) [FMD96]

El sistema es dividido en pequeños y disjuntos subdominios de nodos llamados *ventanas*, i.e., cada nodo pertenece exactamente a una ventana. Se implementa balance de carga en cada ventana mediante comunicaciones estándares. Para poder propagar la carga a todo el sistema la ventana se va "corriendo" ligeramente de manera tal de solapar solamente una parte del viejo dominio para la próxima fase de distribución de carga.

Variación del Algoritmo del Vecindario Directo (X-VD) [FMD96]

Este algoritmo es similar al anterior, su única diferencia reside en que esta variante agrega enlaces a la topología del nodo para formar una topología hipercubo. Este método logra una mejora en la performance respecto de VD para sistemas con mayor cantidad de nodos debido a su naturaleza inherentemente dinámica.

Self-Adjusting Scheduling for Heterogeneous Systems (SASH) [HLA95]

Este algoritmo emplea un *scheduling* completamente solapado que permite que un conjunto de tareas independientes sea asignado a un conjunto de nodos heterogeneos. Este scheduling solapado se logra mediante un nodo S dedicado a ejecutar el algoritmo de scheduling. SASH actúa repitiendo fases de scheduling, creando planificaciones parciales. Al final de cada fase, S envía las nuevas tareas a las colas locales del resto de los nodos.

El algoritmo SASH pertenece a la familia de algoritmos de búsqueda *branch-and-bound*. Realiza búsquedas en el espacio de todas las posibles soluciones (planificaciones) parciales (árbol de subproblemas) para obtener la solución óptima al problema original. Este algoritmo emplea una función de costo para estimar el tiempo de ejecución total que demanda una dada planificación parcial.

En la Tabla 1 pueden observase cómo se clasifican los algoritmos de distribución de carga previamente mencionados. Se observa entonces la utilidad de esta clasificación: simplificar la tarea del diseñador para comparar, desarrollar y seleccionar el mejor algoritmo según sus necesidades y entorno.

5 Conclusiones y Trabajos Futuros

En este trabajo se propuso una clasificación de las estrategias de distribución de carga más general que las propuestas previamente. Además se detalló cómo emplearla para clasificar

Alg. de D.C. / Clasificación	Centralizado	Aleatorio	Vecindario Directo	X-Vecindario Directo	SASH
Invocación	Periódica	Periódica	Periódica	Periódica	Eventos
Ubicación	Centralizado	Dist. Asinc.	Dist. Sinc.	Dist. Sinc.	Centralizado
Decisiones	Global	Local	Local	Local	Global
Comunicación	Física, Dinámica, Global, No Ap.	Física, Dinámica, Global, No Ap.	Física, Fija, Local, No Ap.	Física, Fija, Global, No Ap.	Física, Dinámica, Global, No Ap.
Sel. Nodos	Equipara sobre-cargado con ocioso	Azar	Equipara nodos misma ventana	Equipara nodos adyacentes en hiperc.	Ajusta planif. dinámicamente
Sel. Carga	Buen Candidato	Cualquier Cand.	Cand. Razonable	Candidato Razonable	Buen Candidato

Tabla 1: Clasificación de Algunos Algoritmos de Distribución de Carga.

diferentes algoritmos. Más aún, a partir de este trabajo no es tarea difícil crear nuevos algoritmos según las necesidades de las aplicaciones y el entorno subyacente. Consideramos también que este trabajo puede ser útil al diseñador para comparar diferentes algoritmos entre sí y frente a nuevas propuestas.

A partir de esta clasificación general puede construirse un framework que genere el código fuente del esqueleto de un algoritmo de distribución de carga a partir de la especificación del diseñador. Más aún, este framework podría también incluir librerías que reúnan algoritmos típicos pertenecientes a este campo.

En un futuro próximo se espera diseñar un algoritmo de distribución de carga aplicable al paradigma *process farm*[3] que, luego de una etapa de simulación, constituirá parte integral de un sistema de carga compartida implementado a nivel de kernel bajo Linux[4].

Bibliografía

[BH99] K. Budendorfer y J. H. Hine. A compositional classification for load-balancing algorithms. Reporte Técnico TR-99-9, Victoria University of Wellington, New Zealand, 1999.

[BMD94] K. Benmohammed-Mahieddine y P. M. Dew. A periodic symmetrically-initiated load balancing algorithm for distributed systems. *Proceedings of the ACM SIGOPS*, páginas 66–77, Enero 1994.

[BW88] Katherine M. Baumgartner y Benjamin W. Wah. A global load balancing strategy for a distributed computer system. En *Proceedings of the Workshop on the Future Trends of Distributed Computing Systems in the 1990's*, páginas 93–102, Septiembre 1988.

[BW90] Katherine M. Baumgartner y Benjamin W. Wah. Computer scheduling algorithms: past, present, and future. En *First Workshop on Parallel Processing*, páginas 170–183, 1990.

[CK88] T. L. Casavant y J. G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Trans. on Software Engineering*, SE-14(2):141–154, Febrero 1988.

[DO91] F. Douglis y J. Ousterhout. Transparent process migration: Design alternatives and the Sprite approach. *Software Practice and Experience*, 21(7):1–27, Julio 1991.

[EA03] Javier Echaiz y Jorge Ardenghi. Single System Image: Pilar de los Sistemas de Clustering. *V Workshop de Investigadores en Ciencias de la Computación (WICC 2003)*, páginas 210–214, Mayo 2003.

[3]En este paradigma un proceso maestro ejecuta la parte secuencial de un programa paralelo y dispara (*spawns*) varios procesos esclavos que ejecutarán las porciones paralelas. Cuando un esclavo termina de procesar su carga informa al maestro, el cual le asignará nueva carga. Si bien este paradigma es muy simple, su uso tanto en ambientes académicos como comerciales se encuentra ampliamente difundido.

[4]El kernel de Linux se encuentra disponible, en forma gratuita, en `ftp://ftp.kernel.org`.

[ELZ86] Derek L. Eager, Edward D. Lazowska y John Zahorjan. A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing. *Performance Evaluation*, 6(1):53–68, Marzo 1986.

[FMD96] C. Fonlupt, Ph. Marquet y J.-L. Dekeyser. Analysis of synchronous dynamic load balancing algorithms. En *Parallel Computing: State-of-the-Art and Perspectives, Proceedings of the Conference ParCo'95, 19-22 September 1995, Ghent, Belgium*, volume 11 of *Advances in Parallel Computing*, páginas 455–462, Amsterdam, Febrero 1996. Elsevier, North-Holland.

[Hil85] W. Daniel Hillis. *The Connection Machine*. Series in Artificial Inteligence. MIT Press, Cambridge, MA, 1985. ISBN 0262580977.

[HLA95] B. Hamidzadeh, D. J. Lilja y Y. Atif. Dynamic scheduling techniques for heterogeneous computing systems. *Concurrency: Practice and Experience*, 7(7):633–652, Octubre 1995.

[JM93] C. Jacqmot y E. Milgrom. A systematic approach to load distribution strategies for distributed systems. En *IFIP Transactions: International Conference on Decentralized and Distributed Systems*, Septiembre 1993.

[JV88] W. Joosen y P. Verbaeten. On the use of process migration in distributed systems. Reporte Técnico CW83, Departmente of Computer Science, Katholieke Universiteit, Leuven, Noviembre 1988.

[KL88] P. Krueger y M. Livny. A comparison of preemptive and non-preemptive load distributing. En *8th International Conference on Distributed Computing Systems*, páginas 123–130, Junio 1988.

[Leh93] Morten Lehrmann. Load distribution in Emerald: an experiment. Master's thesis, University of Copenhagen, Junio 1993.

[LM82] M. Livny y M. Melman. Load balancing in homogeneous broadcast distributed systems. En *Proc. Modeling Perform. Eval. Comput. Syst., ACM SIGMETRICS*, páginas 47–55, Abril 1982.

[LMR91] R. Lüling, B. Monien y F. Ramme. Load balancing in large networks: A comparative study. En *Proc. of the 3rd IEEE Symposium on Parallel and Distributed Processing (SPDP'91)*, páginas 686–689, 1991.

[LO86] W. E. Leland y T. J. Ott. Load-balancing heuristics and process behavior. En *Proceedings of the 1986 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Performance Evaluation Review*, páginas 54–69, Raleigh, NC, Mayo 1986.

[OJ92] Kremien O. y Kramer J. Methodical analysis of adaptive load sharing algorithms. En *IEEE Parallel and Distributed Systems*, Noviembre 1992.

[PRR01] A. Plastino, C. Ribeiro y N. Rodriguez. Load balancing algorithms for SPMD applicacions. *Submitted for publication*, 2001.

[Rub87] Harry I. Rubin. The design of a load balancing mechanism for distributed computer systems. Technical Report CSD-87-362, University of California, Berkeley, Julio 1987.

[SS94] Raghu Subramanian y Isaac D. Scherson. An analysis of diffusive load-balancing. En *Proceedings of the 6th Annual Symposium on Parallel Algorithms and Architectures*, páginas 220–225, New York, NY, USA, Junio 1994. ACM Press.

[WM85] Yung-Terng Wang y Robert J. T. Morris. Load sharing in distributed systems. *IEEE Transactions on Computers*, C-34(3):204–217, Marzo 1985.