MONITORIZACIÓN EN TIEMPO REAL DE SISTEMAS INDUSTRIALES DISTRIBUIDOS

Poza Luján, José Luis jopolu@disca.upv.es
Departamento de Informática de Sistemas y Computadoras

Simarro Fernández, Raúl rausifer@upvnet.upv.es
Departamento de Ingeniería de Sistemas y Automática

Simó Ten, José jsimo@disca.upv.es
Departamento de Informática de Sistemas y Computadoras
Universidad Politécnica de Valencia
Camino de Vera s/n - 46022 Valencia

Resumen

La monitorización y detección de errores en los sistemas industriales distribuidos, tiene la dificultad de depender de la topología, tanto física como funcional, del sistema. La gestión de operaciones en una terminal de contenedores, requiere de un conjunto de aplicaciones, que debido a su uso, forman un sistema distribuido. Con el fin de optimizar los costes de las operaciones, toda la gestión de la terminal está automatizada en proporción con la gran complejidad del problema. Este artículo describe el problema desde el punto de vista de la monitorización del estado de la planta y el uso de esta información para la detección de errores y alarmas de situaciones críticas. Se propone una solución a este problema por medio de la monitorización del canal de comunicación de eventos

Palabras Clave: Terminales de contenedores, Sistemas distribuidos industriales, monitorización de sistemas, control de errores, tiempo real.

1. SISTEMAS DISTRIBUIDOS INDUSTRIALES

1.1 INTRODUCCIÓN

Un sistema distribuido se trata de un conjunto de nodos independientes e intercomunicados por una red y provistos de software distribuido, tal que el usuario lo percibe como si se tratase de un único ordenador [1].

En un sistema distribuido la composición y estructura de la red pasa desapercibida al usuario. Solamente le importa los recursos disponibles o, simplemente, su tipo, sin tener en cuenta realmente en qué nodo están ubicados. El objetivo primordial de los sistemas distribuidos es el comportamiento fácil y eficiente de los recursos entre múltiples usuarios o aplicaciones.

1.2 CARACTERÍSTICAS

Las principales características de un sistema distribuido, que los diferencia de uno centralizado, son las siguientes:

- **Distribución de componentes** en los nodos de una red.
- **Heterogeneidad** : los componentes pueden estar constituidos con tecnologías diferentes a sus distintos niveles: hardware, Sistemas Operativos, redes de comunicación y protocolos, lenguajes de programación, aplicaciones,…
- **Dificultad para determinar el estado global** : el estado global de todos sus componentes en un instante dado (instantánea o *snapshot*) es difícil de obtener debido a la débil sincronización.
- **Fallos parciales** : los modos de fallo de los componentes son independientes entre sí.

1.3 DISEÑO

La división de un sistema en subsistemas, la encapsulación de los subsistemas, la conservación de los estados en caso de fallo, y lo que es más importante, un estricto control sobre el modelo de interacciones en los subsistemas, son la clave para controlar la complejidad de un gran sistema [2].

1.3.1 Características

Las **reglas básicas** para el diseño de aplicaciones distribuidas son:

- La **descentralización** : evitar soluciones donde un único componente tiene información completa sobre el estado del sistema. Ello reduce:

- La **escalabilidad** : las prestaciones se degradan conforme aumenta el número de clientes.
- La **tolerancia a fallos** : el fallo de un componente deja inoperante al sistema.
- **Replicación** (total o parcial) : mantener la información redundante con el fin de mejorar:
- La **autonomía** : la información se mantiene replicada en los nodos donde se utiliza con el fin de minimizar los accesos a la red.
- La **disponibilidad** : seguir proporcionando un servicio en presencia de fallos.

1.3.2 Objetivos del diseño.

- **Interoperabilidad** : integración de los sistemas y componentes heterogéneos en un "todo" sin costosos desarrollos personalizados.
- **Transparencia** : ocultación de la distribución al usuario. Un nivel de transparencia total suele tener un coste demasiado elevado.
- **Carácter abierto / modularidad**: permitir ampliar la funcionalidad del sistema con nuevos componentes a través de interfaces claras.

2. MONITORIZACION

2.1 SISTEMAS DE MONITORIZACIÓN.

Para la obtención de las variables características del funcionamiento del sistema tales como envío de mensajes entre aplicaciones; o los índices de prestaciones tales como tiempos de proceso, se utilizan los denominados monitores. Los monitores observan el comportamiento del sistema en unas determinadas condiciones de carga durante un cierto intervalo de tiempo conocido como sesión de medida. Para abordar la monitorización de un sistema [3], se deben tener claras algunas cuestiones tales como:

- **Selección**: se debe que ser lo más selectivo posible debido a que se corre el riesgo de extraer una información excesiva o redundante que únicamente dificultará cualquier labor de estudio.

- **Localización**: situación de dónde se encuentra la información, lo cual orienta en la elección del tipo de monitor a emplear.

- **Análisis**: cómo se puede extraer la información, sabiendo la forma de extraer la información, no solo se sabe cómo desarrollar la herramienta, sino que también cómo diseñar el experimento.

Todas las herramientas de monitorización se pueden clasificar conceptualmente en tres bloques principales, un **sensor** o *elemento lector* que se encarga de recoger la información necesaria; el **transformador** o *elemento de proceso* que procesa la información recogida y un *elemento interprete* o **indicador** que muestra el resultado de forma entendible. Para clasificar, y estudiar, los monitores de sistemas, se organizan dependiendo de los niveles sobre los que actúan, tal como se ve en la figura 1.

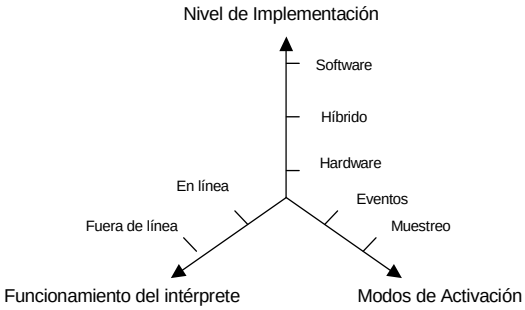


Figura 1. Clasificación sistemas de monitorización.

Nivel de implementación. Define a qué nivel del sistema está implementado el monitor. El nivel software es el compuesto de uno o varios programas, que por medio de rutinas monitorizan la actividad del sistema. El nivel hardware se implementa por medio de instrumentos conectados al sistema dedicados expresamente a ello. El monitor híbrido consiste en un monitor hardware que funciona con el apoyo de un monitor software.

Modos de activación. Determina la forma en que se va a poner en marcha la actividad del monitor. Puede ser de dos tipos: eventos y muestreo. En el modo de detección de eventos es aquel en el que dependiendo del estado del sistema el monitor entra en funcionamiento (por ejemplo ante lecturas de memoria, accesos a disco, etc.). En el modo de muestreo, la intervención del monitor la provoca un temporizador o bien está determinada por unos intervalos constantes de tiempo o eventos periódicos.

Funcionamiento del elemento intérprete. Especifica cómo actúa el interprete del monitor. En el intérprete en línea, el elemento funciona a la vez que se desarrolla la sesión de medida del monitor. En el intérprete fuera de línea, se actúa a la conclusión de la fase de monitorización.

2.2 CONTADORES

Habitualmente conocidos como contadores de rendimiento, los contadores son la forma más habitual de monitorizar la evolución de un sistema.

El problema de los contadores es la escasa posibilidad de detección de estados problemáticos a parir de información cuantitativa. Por ello, los

contadores se emplean, mayoritariamente, en la monitorización de parámetros de sistemas operativos [11].

Debido a la necesidad de información cualitativa en la inferencia de situaciones problemática o análisis de estados, no son el método más adecuado para la monitorización de sistemas industriales complejos.

2.3 GENERACIÓN DE TRAZAS

Una traza es una descripción real de la evolución temporal de un sistema. Para generar esta descripción se debe disponer de acceso a los parámetros del sistema que sean relevantes.

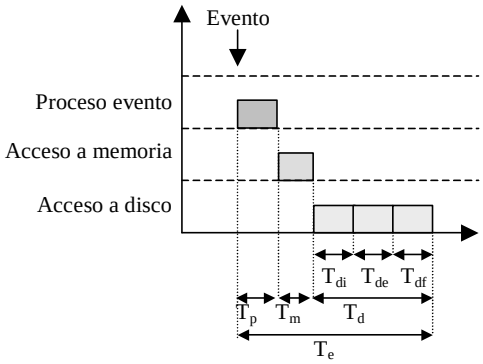
La generación de trazas se ha realizado generalmente por medio de librerías o unidades dedicadas a ese efecto, tal como se presenta en [5]. En ocasiones, estas unidades deben combinarse con el código del programa o aplicación a monitorizar y en otras puede hacerse por medio de una conexión al canal de comunicación o al sistema hardware. Tanto de una forma como de otra, la traza deberá mantener la correspondiente coherencia y homogeneidad en su creación. En los sistemas distribuidos de tiempo real, el uso de estándares POSIX en la traza está siendo actualmente muy estudiado [10]. Mantener la correspondiente coherencia con tiempo real se hace necesaria, de ahí que en los siguientes puntos se

La monitorización de sistemas no deja de ser un método que pueda interferir en el funcionamiento del sistema. En los sistemas distribuidos en tiempo real, esta interferencia de la observación puede resultar problemática, dado que la generación de la traza y el almacenado en un archivo es una tarea más en el sistema.

Las tareas implicadas en un evento que se deben observar son tres:

- **Gestión del evento**. Compuesta por los cálculos asociados y creación de las cadenas de caracteres de la traza.
- **Acceso a memoria**. Consistente en el almacenamiento en la memoria del procesador de la cadena de traza generada anteriormente.
- **Acceso a disco**. Es la tarea en la que se almacena en un archivo de disco (traza) la cadena de caracteres correspondiente a la monitorización.

La secuencia de las tareas implicadas para la generación de la traza de un solo evento, es la que se observa en la figura 2.



T_e Tiempo total de proceso del evento.
T_p Tiempo de procesamiento del evento
T_m Tiempo de almacenamiento en memoria de la traza
T_d Tiempo de almacenamiento en disco de la traza
T_{di} Tiempo de disco inicial (apertura y posición en el archivo)
T_{de} Tiempo de acceso al disco para almacenado de traza
T_{df} Tiempo de disco final (cierre del archivo)

Figura 2. Tareas de generación de una traza implicadas en un evento.

La gestión de las tareas implicadas en la generación del log es importante, por ello se verán más detenidamente a continuación.

2.3.1 Generación simple

La aproximación más básica de la monitorización se hace cuando los eventos de la aplicación que deben ser monitorizados, son almacenados en una cadena, que al finalizar la aplicación, se almacenará en el archivo de log correspondiente (figura 3).

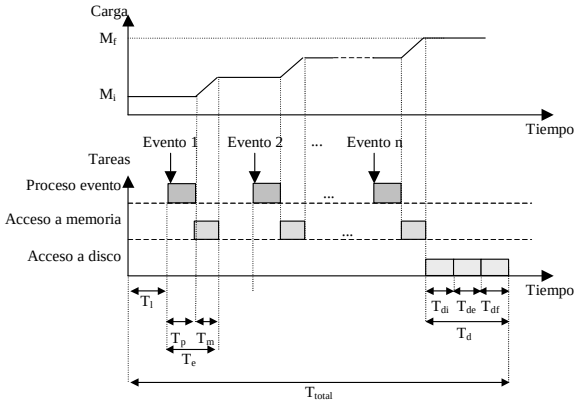


Figura 3. Esquema de generación simple.

Este tipo de generación, tiene varios problemas que no compensan la sencillez del planteamiento. Los principales problemas que se pueden observar son:

- No se puede acceder al archivo de log mientras la aplicación no haya guardado el archivo de log.

- La cadena de log, se debe almacenar en memoria, esto no sólo implica una pérdida de la misma, sino que puede llegar a saturar y provocar el mal funcionamiento de la aplicación.

Este modo de generación de trazas sólo se emplea en los ensayos, donde la duración de una sesión de muestreo es limitada.

2.3.2 Generación directa a disco.

Una de las posibles soluciones que eviten la sobrecarga de la cadena de almacenamiento, pasa por abrir el archivo sólo cuando se precise escribir la cadena del log. Este tipo de monitorización tiene el problema del tiempo de apertura y cierre de un archivo. Este tiempo, en comparación con el procesamiento interno de la aplicación es elevado y puede influir en el rendimiento de la misma. Sin embargo, el archivo de log permite ser copiado y analizado *on-line*, por lo que este modo de generación es adecuado en las simulaciones extensas del sistema.

Para solucionar el problema de la memoria ocupada por la cadena y del tiempo de apertura y cierre del archivo, se puede almacenar en el archivo de *log* los resultados en cuanto se generan, es decir, se tendrá un archivo de log abierto en cuanto se inicie la monitorización, y este archivo se cerrará en cuanto finalice la misma

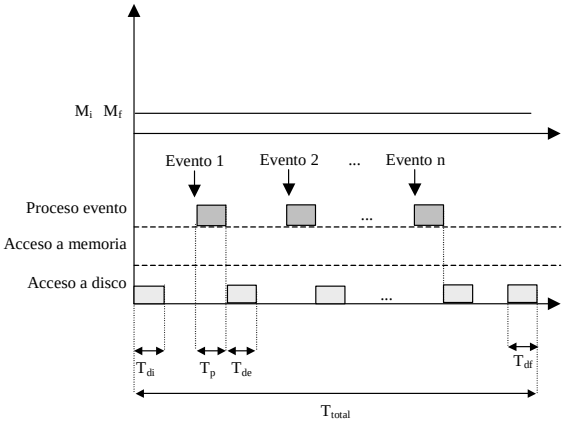


Figura 4. Esquema de generación directa a disco.

Este sistema, evita el posible desbordamiento de la memoria por parte de la cadena de log, pero introduce algún que otro problema al tener un enlace al archivo abierto. Esto implica que, en el momento en que varias aplicaciones quieran compartir el archivo tendrán problemas por el intento de apertura. Aunque eso se puede gestionar internamente, tampoco es una situación recomendable. Esta situación se da cuando la aplicación crea hilos que comparten el log. Estos hilos no pueden acceder al archivo a la vez si no es por medio de colas controladas por semáforos, por lo que el archivo puede no estar actualizado.

2.3.3 Generación Híbrida

Un esquema de monitorización interesante que se puede implementar es el híbrido. Este sistema es el que une las filosofías vistas en los dos últimos apartados, es decir, debe evitar el uso excesivo de memoria por parte de la cadena de log y no debe ralentizar la ejecución normal del sistema a causa de la escritura del archivo en disco (figura 5).

Este esquema implica la creación de un buffer de mensajes que, al alcanzar cierto tamaño, se almacena en el archivo, de esta forma nos aseguramos que no rebasamos el tamaño máximo de la cadena de log, ni que perdemos el tiempo en abrir, acceder o cerrar archivos, aparte de tener en cierta medida el archivo de log actualizado cada cierto intervalo de tiempo.

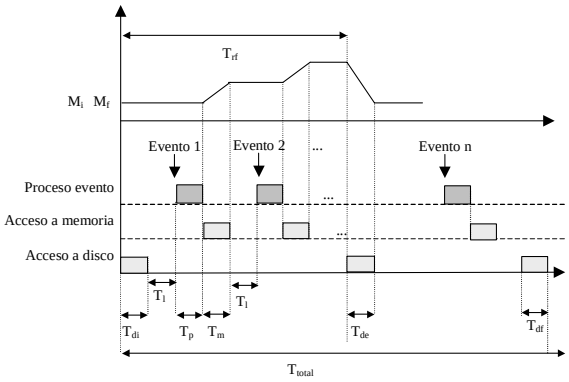


Figura 5. Esquema de generación híbrida.

En este modelo tenemos dos parámetros para determinar la eficiencia de la monitorización, uno es el tamaño del buffer de la cadena del log, y otro es la frecuencia de actualización del archivo. El balanceado de cada uno de los parámetros, puede determinar la forma óptima de generar un log, sin que se interfiera en las condiciones de tiempo real del sistema.

2.3.4 Formato de la traza

El archivo de Log contiene la información relativa al sistema que se ha decidido almacenar. La organización interna de dicho archivo puede estar orientada a líneas , a columnas o a registro. La organización física de los datos condiciona en gran medida las posibilidades de análisis de los mismos [6].

2.4 LOCALIZACIÓN DE LAS TRAZAS

2.4.1 Trazas distribuidas

Las trazas distribuidas son aquellas en las que cada nodo genera un archivo de log (figura 6). Si bien, el análisis implica el envío, o el acceso, a cada uno de los archivos.

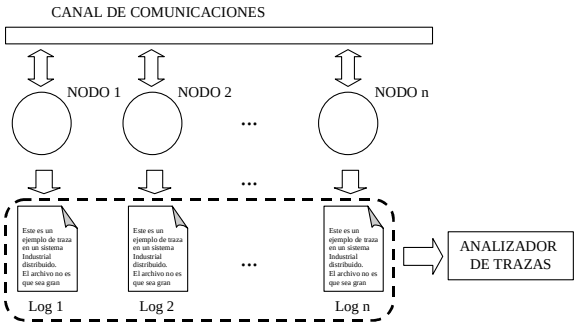


Figura 6. Trazas distribuidas.

Este acceso a los archivos añade una pérdida de tiempo importante, puesto que supone ocupar el canal de datos del sistema para transmitir información.

Este modo de trabajo no se emplea habitualmente para una monitorización de estado, errores o de alarmas. Sin embargo, es muy apropiada para realizar informes de funcionamiento en los que el tiempo no sea crítico.

2.4.2 Traza centralizada.

La traza centralizada se genera en un nodo conectado directamente al canal de eventos (figura 7).

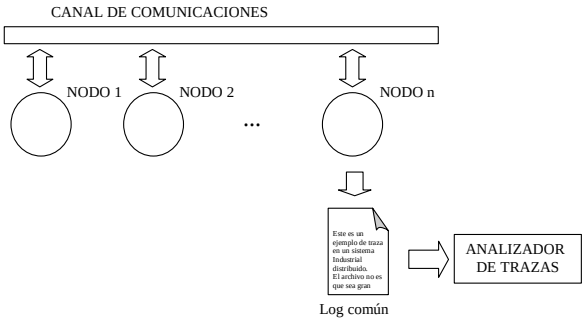


Figura 7. Traza centralizada.

La generación de trazas orientada al canal de comunicaciones, en lugar de la traza orientada a las aplicaciones, ofrece la posibilidad de conocer los eventos ocurridos en el sistema en el mismo instante en el que se dan. Además, el hecho de evitar la transmisión de archivos por canales de datos, libera a estos de una carga importante.

3. SISTEMA ACTUAL

3.1 TERMINALES DE CONTENEDORES

Los contenedores son un medio ampliamente utilizado en el transporte de mercancías debido al servicio puerta-puerta que ofrecen y la rapidez en que puede ser realizadas las operaciones entre el servicio marítimo de transporte y el terrestre. Estas ventajas han motivado que el volumen de transporte marítimo de contenedores haya aumentado considerablemente en los últimos años y en particular, en la Comunidad Valenciana [7].

En una terminal marítima de contenedores se realizan las operaciones de carga y descarga de los mismos en los barcos y sirve de punto de interconexión del transporte marítimo con el terrestre (camiones y ferrocarril). Para realizar este proceso es necesario almacenar temporalmente los contenedores hasta que son despachados a su destino utilizando máquinas especializadas en su manipulación (transtainers, frontales, grúas porta contenedores, etc.), tal como se observa en la figura 8.

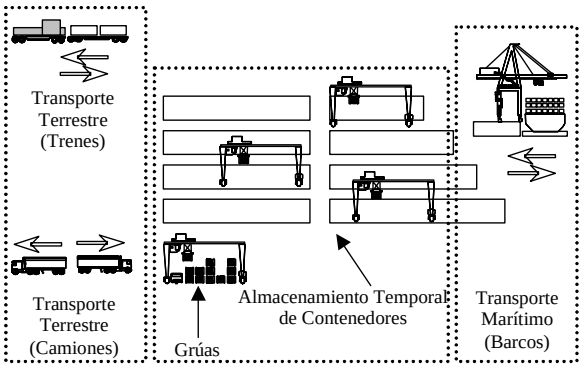


Figura 8. Operativa de la terminal de contenedores.

Estas terminales normalmente son grandes y, debido al impacto medioambiental de los grandes puertos y sus grandes costes, es necesario optimizar la utilización de este espacio y los recursos implicados en la gestión de los contenedores [8].

3.2 GAMA

GAMA (Gestión Automática de una terminal Marítima de contenedores) desarrolla un sistema de automatización global de la terminal que determina y planifica en tiempo real las actuaciones necesarias para la mejor explotación, ubicación de los contenedores, gestión de puertas, planificación de la estiba, gestión de máquinas y otras tareas, a partir de la configuración física de la terminal, las necesidades y los recursos disponibles en cada momento.

Para la consecución de este objetivo, se han identificado unas áreas de actuación concretas sobre las que descansa la estructura del sistema global. Los sistemas que implementan cada una de las áreas de interés sobre las que se centra GAMA se verán a continuación.

3.2.1 Sistema de ubicación de contenedores.

Se encargan de la optimización de la ubicación de contenedores. Esta aplicación tiene como objetivo generar las posiciones donde deben ubicarse los contenedores. Este módulo, devuelve la posición donde se desea que se sitúe un contenedor de entrada a la terminal teniendo en cuenta por dónde entra (puertas, buque y ferrocarril) y las características del mismo (servicio, destino, peso, ...).

3.2.2 Sistema de gestión de máquinas

Esta aplicación es la encargada de realizar la asignación de máquinas para ubicar, desubicar y remocionar contenedores de la terminal. En función de la hora del día, determina el número de máquinas operativas en la terminal, asigna el rango de calles y pilas donde deben operar y el tipo de operativa (marítima o terrestre). Así mismo, este módulo también es el encargado de realizar el paso de mensajes entre las máquinas y de la actualización de la base de datos del estado de las máquinas.

3.2.3 Sistema de información y visualización de la terminal.

Con este sistema se consigue de forma gráfica, la visualización de los contenedores y las máquinas que se encuentran en la terminal en cada instante. Permitiendo a cada operador una visión de la situación de los contenedores que se encuentran en la base de datos y de las posiciones que van enviando las máquinas de la terminal

3.2.4. Sistema de Control de accesos.

Este módulo es el encargado de la asignación de los dispositivos TAG's (etiquetas electrónicas) a los camiones de entrada a la terminal, y que permiten la identificación de los camiones durante su estancia. Así mismo, esta aplicación actualiza la base de datos cuando el camión abandona la terminal.

3.3 COMUNICACIONES

Con la distribución de las aplicaciones lo que se logra es la descentralización en la toma de decisiones. Para ello, lo que existe son unos canales de comunicación que garantizan el envío y la recepción de mensajes entre aplicaciones (comunicación fluida y segura).

En el sistema GAMA existen dos "canales" para comunicar las aplicaciones, el canal de datos y el canal de eventos. El canal de datos es el que utilizan las aplicaciones para consultar y actualizar la base datos. Este canal se desarrollo por medio del interfaz ODBC (Open DataBase Connectivity). El canal de eventos es el que utilizan las distintas aplicaciones para realizar el paso de mensajes de forma síncrona.

El soporte de datos en Marítima Valenciana S.A. se encuentra en un AS/400, el cual ofrece el protocolo ODBC, que como se ha comentado anteriormente es el que se utiliza para consultar y actualizar la información. La base de datos es la que mantiene la información actualizada con los contenedores que se encuentran en la terminal, la posición de las máquinas y sus características, las entradas y salidas de buques, trenes, camiones y otras máquinas. En definitiva, es donde se encuentra el estado de la terminal y donde todas las aplicaciones consultan y actualizan la información. La actualización de un campo en esta base de datos genera un evento que informa al resto de las aplicaciones.

Debido a ese flujo de información que se produce entre las aplicaciones y a la distribución de las mismas, se hace imprescindible un sistema que monitorice y administre el flujo de mensajes. Para poder controlar el sistema y dar el soporte a la tolerancia a fallos.

Para permitir la interacción entre procesos de Win32 en el sistema distribuido, se emplea un interface común de comunicación a nivel de eventos llamado Servidor de Comunicaciones (SC), mostrado en la figura 9. El SC se basa en la representación interna de datos por medio de una pizarra distribuida.

La estructura de datos representada en la pizarra está continuamente actualizada por el SC, mostrando los valores de los mensajes entre los procesos. A través del acceso a la estructura de datos del SC, los procesos distribuidos se comunican usando un interfaz común independientemente de su ubicación (figura 9).

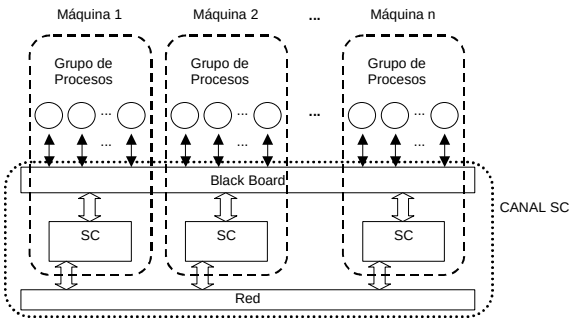


Figura 9. Sistema de comunicaciones SC

El sistema SC requiere que cada computador que se encuentre en el sistema ejecute una instancia del SC. Estas instancias son las que se encargan de realizar las labores de comunicación, actualizando y controlando los datos para obtener, de esta forma, copias parciales de la pizarra en distintos ordenadores tal como se muestra en la figura 9. Esto hace que los procesos, independientemente de su localización, sólo deban acceder a la correspondiente instancia del SC para contactar con el resto del sistema

3.4 PROBLEMÁTICA

Este sistema debe permitir que cada aplicación pueda encontrarse en un nodo distinto de la red, esta posibilidad facilita el manejo a los operadores que pueden trabajar cada uno desde su terminal, pero dificulta la determinación del estado global.

El sistema GAMA se encuentra constituido por aplicaciones creadas con lenguajes de programación distintos (Visual C++, MODSIM,...), las máquinas utilizan sistemas operativos heterogéneos : el soporte de datos se encuentra en un AS/400, mientras que las aplicaciones se ejecutan sobre Windows2000, el hardware es distinto para cada nodo de la red y otros aspectos.

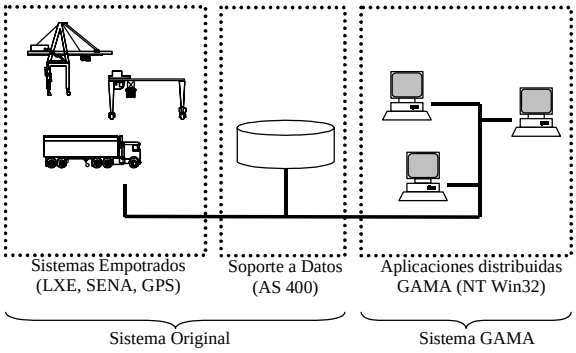


Figura 10. Conexión de GAMA al sistema actual.

El fallo de cualquiera de las aplicaciones provoca un mal funcionamiento del sistema, por este motivo se hace necesaria la monitorización del sistema mediante la generación de trazas que se propone en este artículo.

3.5 SOLUCIONES

La identificación de las áreas de interés del problema, y la aplicación que lo resuelve, permite la posibilidad de añadir o reducir las aplicaciones para adaptarse a las tareas concretas que se estén tratando, y permite que nuevas aplicaciones se conecten al sistema o actualicen sin necesidad de reconfigurar el resto de módulos[2].

El problema de la tolerancia a fallos queda solucionado mediante la generación de trazas particulares para cada aplicación y de una aplicación que gestiona la traza global que permite la monitorización del sistema en conjunto.

El SC y el uso de las pasarelas de comunicación hacen al usuario de cada aplicación GAMA transparente en la transferencia de mensajes con el resto de módulos, ofreciendo la posibilidad de monitorizar al canal en lugar de a las aplicaciones.

4. SISTEMA DESARROLLADO

4.1 NIVELES DE MONITORIZACIÓN

La captura de eventos puede realizarse a tres niveles, dependiendo de la importancia:

1. Estado.
2. Errores.
3. Alarmas.

En el nivel de captura de estado, el monitor se conecta a todos los datos compartidos por las aplicaciones y, tras un filtrado previo, se visualiza la actividad en cada uno de los módulos de GAMA.

La visualización del estado, se puede hacer por muestreo constante u orientada a los eventos del sistema. Si la visualización es por muestreo, se puede obtener una copia de la situación actual en el canal de comunicación, esta forma de trabajo no es la habitual. Si la visualización se hace por eventos del canal, se obtiene un registro secuencial de la actividad del mismo o lo que equivalente, se puede seguir la ruta de los mensajes por los distintos nodos del sistema.

La visualización del estado es importante para llevar un control de la actividad normal del sistema, comprobar los cambios de estado correspondientes y tareas similares. Sin embargo, añade un nivel de carga en el canal de monitorización que no es deseable, aunque se haya constatado que su efecto es casi nulo sobre el funcionamiento normal.

En el nivel de comprobación de errores no críticos, se monitorizan únicamente datos del canal de comunicaciones orientados a errores. Las aplicaciones distribuidas informan a estos datos de los errores, tales como el time-out de un mensaje, error de formato en contenido, repetición de mensajes o errores similares. Este nivel de monitorización se emplea para comprobar la coherencia en las comunicaciones.

Las alarmas son aquellos errores que las aplicaciones detecten y sean significativos para el funcionamiento global del sistema. Errores como la ausencia de un

nodo básico, la caída de una pasarela, o situaciones críticas como la posibilidad de un accidente en planta justifican este canal de máxima prioridad.

4.2 BLOQUES DE LA APLICACIÓN

La división en bloques de la aplicación se hace en base a la funcionalidad de los mismos [5], los bloques del sistema de monitorización se pueden ver en la figura 11.

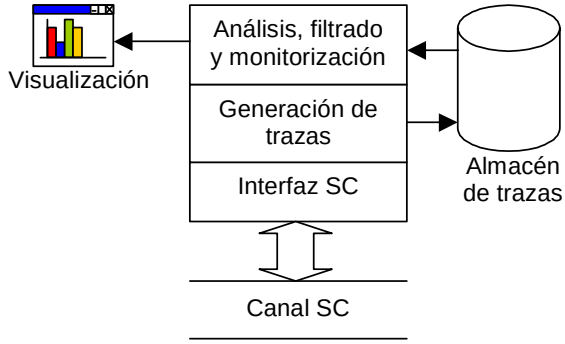


Figura 11. Bloques de monitorización.

No necesariamente estos bloques deben ser parte de una misma aplicación. A continuación se observará con más detalle los bloques mencionados anteriormente.

4.3 INTERFAZ SC

La interconexión del sistema de monitorización con el canal SC se hace en función de las variables que se deseen visualizar o que sean relevantes.

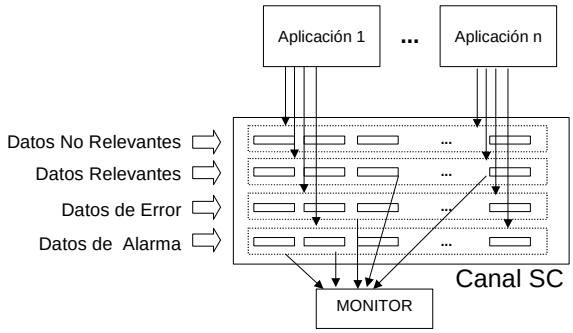


Figura 12. Interconexión con SC.

Cabe destacar que la conexión a los datos del canal SC por parte de las aplicaciones se hace desde un punto de vista vertical, en el que cada aplicación se conecta a los distintos niveles en los que adquiere relevancia.

En el caso del monitor, la conexión se hace desde un punto de vista horizontal, en el que se conectan datos dispersos de un mismo nivel de comunicación.

La conexión con los datos puede ser selectiva y configurable, por lo que en cualquier momento es posible obtener la información concreta de la planta o de los errores producidos y variar esta visión

4.4 GENERACIÓN DE TRAZAS

4.4.1 Clase generadora

La generación de trazas se realiza por medio de la clase CLog desarrollada en C++ con el soporte de la MFC de Visual C++ 6.0. El empleo de esta herramienta se debe a varios factores, entre ellos al soporte a hilos que proporciona, la velocidad de ejecución, y la gran capacidad de integración y reciclaje de código por medio de la generación de librerías de enlace dinámico (DLL) o la compartición de objetos por DCOM.

El uso de una clase común por parte de las distintas aplicaciones implicará que todas ellas compartan la misma estructura del archivo de log. Esto es fundamental a la hora de mantener una compatibilidad horizontal entre aplicaciones y el análisis del comportamiento de las mismas.

En la creación del objeto correspondiente al log que genere el monitor, se pueden configurar los siguientes parámetros:

- **Archivo y ruta del log**. Parámetros clásicos del archivo de log a generar.
- **Modo en que se almacena el archivo en disco**: creación de un nuevo archivo (con nombre distinto en cada creación), eliminado del anterior log o añadido al log ya existente.
- **Modo en que se almacena la cadena de memoria** en el archivo: final, continuo o fragmentado.
- **Modo en que se sincronizará el almacenado** del log: directo, a intervalos de tiempo o al final de la sesión.

Las distintas combinaciones de los dos últimos parámetros dan lugar a las distintas políticas de generación del log.

4.4.2 Formato de archivo

La definición del formato de cada elemento mínimo de información del log se hace en la creación del objeto.

En la versión actual la información del archivo está orientada a línea y a columnas dentro de cada línea tal como se ve en la figura 13. Esto se debe al funcionamiento orientado a filas y columnas de EXCEL, y que es de gran utilidad para el análisis de las trazas.

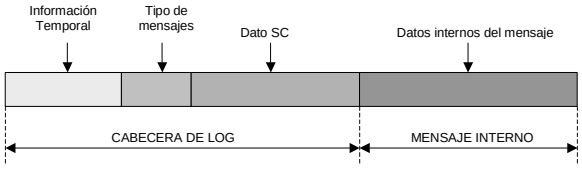


Figura 13. Formato de línea del log.

El campo de información temporal se emplea para mantener el control del tiempo en la gestión de mensajes, este campo puede contener una fecha, una hora o información temporal de control del sistema distribuido de los nodos por medio del protocolo TTP [1] o similar de control del tiempo.

El campo correspondiente al tipo de mensajes, se emplea para el almacenamiento del tipo de evento que se ha producido (evento, error o alarma). Este campo puede tomar valores que permitirán el posterior filtrado de la información.

El campo correspondiente al dato SC, permite conocer qué aplicaciones del sistema distribuido están implicadas en la secuencia de eventos. Los datos SC son actualizados por aplicaciones, esto hace que de una actualización y del posterior disparo de la monitorización se pueda deducir qué aplicación o aplicaciones están implicadas en el evento.

El campo del mensaje interno, permite conocer el estado de la planta. Este campo contiene información de cómo está evolucionando la planta. Por ejemplo, las máquinas almacenan en este campo posiciones, modos de actuación e información similar, por lo que a partir del contenido del mismo se puede deducir movimientos de máquina a lo largo de actuaciones.

El formato del archivo es configurable, sin embargo es muy importante que sea común para cada tipo de análisis.

4.5 ANÁLISIS Y FILTRADO.

El análisis y el filtrado del archivo de log se puede realizar por medio de la aplicación de cálculo EXCEL® de Microsoft®.

La versatilidad en la creación del archivo de log, hacen que no sea necesaria la existencia de un módulo específico a nivel de aplicación.

Sin embargo, las posibilidades de análisis que ofrece un archivo con líneas orientadas a campos y con una sintaxis bien conocida de separación de campos y contenidos, hace que la exportación de datos hacia otras aplicaciones sea muy simple, por lo que para análisis sencillos y filtrado de datos, es muy eficiente el uso de estas aplicaciones.

5. PRUEBAS REALIZADAS

Como ejemplo de traza generada se ha monitorizado la evolución temporal de dos tipos de mensajes el EVENASICONTENEDOR, perteneciente al grupo de mensajes de eventos de contenedores (EC), que nos proporciona los eventos de un contenedor a lo largo de su estancia en la terminal y el EVENASITRANSTAINER, perteneciente al grupo de mensajes de eventos de máquinas (EM) que proporciona los eventos que una máquina genera por su actividad en la terminal.

El resultado se muestra en la figura 14. Se puede apreciar cómo la generación del mensaje con un formato conocido, es de gran utilidad para poder separar la información en columnas. Esta separación facilita enormemente el análisis de los mensajes.

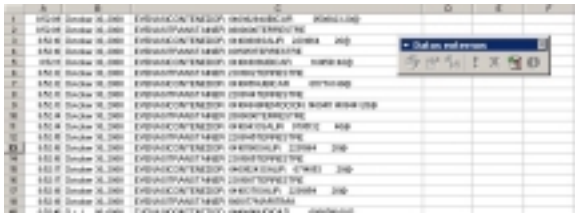


Figura 14: Aspecto de la hoja de Excel representando una traza.

El análisis gráfico expuesto en la figura 15 de la traza mostrada en la figura 14 da a entender una gran descompensación entre los mensajes relativos a contenedores en relación a los mensajes relativos a las máquinas. Esto quiere decir que la ocupación del canal de comunicaciones por parte de las máquinas es mayor que los contenedores.

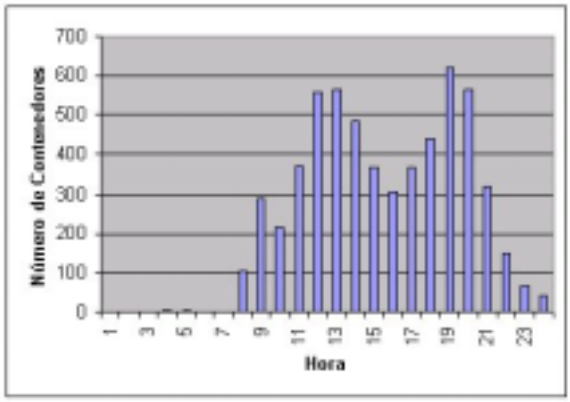


Figura 15: Gráfico comparativo de la frecuencia de eventos en función de la hora, para un día concreto.

A partir de los resultados expuestos anteriormente, se puede determinar algunos aspectos como la prioridad

de los mensajes relativos a contenedores o el ancho de banda dedicado a los mensajes de máquina dependientes de la hora en la que se pueda prever el aumento de tráfico.

También es posible determinar algunos patrones de comportamiento que faciliten el establecimiento de políticas de ajuste del sistema distribuido en función de la carga que tenga en cada momento.

6. CONCLUSIONES

En las primeras fases del diseño de sistemas, es importante la capacidad de análisis de las características del mismo. Aquí se ha propuesto un método orientado a monitorizar, de forma sencilla, un sistema distribuido. El análisis se realiza por medio de una herramienta comercial genérica. Esto puede implicar limitaciones en el ámbito de los resultados. Sin embargo, el nivel de concreción que se debe obtener de los datos, no debe ser tan elevado como para precisar de herramientas avanzadas de análisis.

Actualmente, está en fase de desarrollo una versión de la clase CLog con posibilidades de generación de trazas en archivos de base de datos SQL Server. Este aspecto tiene un gran interés, dado que la conexión de SQL Server con las aplicaciones GAMA o con EXCEL por medio de ODBC es muy sencilla, esta facilidad aumenta considerablemente la potencia de análisis de trazas al poder realizar consultas desde cualquier aplicación y en cualquier momento de la generación de la traza.

El interés de realizar consultas en línea desde una aplicación es muy alto, puesto que permite obtener una visión instantánea del funcionamiento y la carga del sistema, o lo que es lo mismo, del rendimiento actual. Sobre estos parámetros se puede inferir el comportamiento del sistema y tomar decisiones para balancear, por medio de prioridades, el uso que las aplicaciones o las máquinas hacen de los canales de comunicación.

Agradecimientos

Este trabajo está financiado por la CICYT y la Comisión Europea a través del proyecto FEDER-CICYT número 1FD97-2158-C04-0X y por la empresa Marítima Valenciana S.A.

Referencias

[1] Kopetz H. (1998). The Time-Triggered Model of Computation. 0-8186-9212-X/98 *IEEE*.

[2] Kopetz, H., (1997) *Real-Time Systems : Design Principles for Distributed Embedded Applications*, pp 29-44.

[3] Mansouri, M. Sloman, M. *Monitoring Distributed Systems (A survey)*. Imperial College Research Report No. DOC92/23. September 1992.

[4] Meyer, B. Heineken, M., Popien, C.. *Performance Analysis of Distributed applications with ANSAmon*.

[5] Ohlenroth, M., (1996) *Application Oriented Monitoring*, TUCZ / RA-TR-96-09.

[6] Ramakrishna, Raghu. "Database Management Systems". McGraw-Hill, 1998.

[7] Ruiz Vergara, L. *Memoria de actividades de la Compañía Marítima Valencia S.A.* Valencia 1999.

[8] Simarro, R., J.L Navarro, I. Huet, E. Orellana. (2001). *Simulación de una terminal marítima de contenedores*. Workshop en Metodologías de Modelado y Simulación de Sistemas. Barcelona.

[9] Tanenbaum A.S. *Distributed Operating Systems*. Prentice Hall International. 1995.

[10] 1003.1q-2000. *Standard for Information technology-Portable Operating Systems Interface (POSIX®)-Part 1: System Application Program Interface (API)-Amendment 7: Tracing [C Language]*.

[11] Dennis C. Lee, Patrick J. Crowley, Baer J.L., Thomas E. Anderson. *Execution Characteristics of Desktop Applications on Windows NT*. ISCA 1998