Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXVII, págs. 1063-1084 (artículo completo)
Alberto Cardona, Mario Storti, Carlos Zuppa. (Eds.)
San Luis, Argentina, 10-13 Noviembre 2008

ESTUDIOS PARAMÉTRICOS DE MECÁNICA DE SÓLIDOS EN ENTORNOS DE COMPUTACIÓN DISTRIBUIDA

Carlos Catania [a], Claudio Careglio [a,b,c] , David Monge [a,c] , Paula Martinez [a], Anibal Mirasso [a,b] , Carlos Garcia Garino [a,b]

[a]LAPIC, ITIC, Instituto Tecnológico Universitario & Universidad Nacional de Cuyo
Casilla de Correo 947, 5500 Mendoza

[b]Facultad de Ingeniería, Universidad Nacional de Cuyo,
Centro Universitario, 5500 Mendoza

[c]Becario Doctoral, CONICET
{ccatania, pmart, dmonge, cgarcia}@itu.uncu.edu.ar; {ccareglio, aemirasso}@uncu.edu.ar

Palabras Clave: Mecánica de Sólidos, Computación Distribuida, Estudios Paramétricos, Condor.

Resumen. Actualmente se cuenta con distintos métodos de computación distribuida capaces de incrementar los recursos de cálculos y/o disminuir tiempos de ejecución de aplicaciones intensivas, como resultan algunos casos de elementos finitos.
En este sentido se emplean cada vez más frecuentemente códigos paralelos que se procesan en distintas computadoras a partir de bibliotecas como Message Passing Interface (MPI), para lo cual se reparte el dominio de aplicación en diferentes computadoras. Sin embargo existen otras posibilidades de procesos concurrentes como es el caso de estudios paramétricos, en donde se debe ejecutar un gran número de problemas con el mismo código y diferentes datos. Este tipo de problema, que se conoce como inherentemente paralelo, puede ejecutarse con relativa facilidad en entornos de computación de alta disponibilidad como el middleware Condor.
Condor es un entorno de trabajo multiplataforma que permite el procesamiento de tareas mediante la gestión de recursos ociosos. Entre sus características principales se destaca la flexibilidad para la gestión de trabajos, seguridad, posibilidad de ejecución condicional, checkpointing, etc.
El presente trabajo discute la ejecución paralela del código de elementos finitos SOGDE en el entorno Condor. Para instrumentar la ejecución de SOGDE sobre Condor se ha diseñado una aplicación administradora que se encarga de la orquestación del flujo de datos y tareas. La misma efectúa las tareas para generar automáticamente los ficheros de datos, procesar los mismos mediante el código SOGDE y recuperar resultados para su correspondiente edición y post-proceso, mediante herramientas adecuadas como gnuplot o similares.
De esta forma pueden realizarse estudios paramétricos mediante una ejecución única que varía los datos en forma automática y entrega un informe de resultados, listado de tablas, gráficos, etc.
El uso de la aplicación se ejemplifica mediante estudios de sensibilidad a imperfecciones de un problema de estabilidad elástica bidimensional (2D), para lo cual se escoge el caso de la columna de Euler. Obtener los resultados esperados requiere procesar un gran número de problemas no lineales con SOGDE.

1 INTRODUCCIÓN

El presente trabajo discute el empleo del middleware Condor (Thain, Tannenbaum y Livny, 2002; Livny, Basney, Raman y Tannenbaum, 1997) en el contexto de Mecánica Computacional, con el objetivo de llevar a cabo estudios paramétricos de manera concurrente y automática. Para procesar los problemas de Mecánica Computacional se utiliza el código SOGDE (García Garino, 1993; García Garino y Oliver, 1995; García Garino y Oliver, 1996).

Un estudio paramétrico de un problema de interés requiere variar de manera adecuada los datos del problema (geometría, materiales, imperfecciones, etc.) y procesar luego diferentes problemas con cada juego de datos. Si bien es posible realizar manualmente este tipo de estudios es una tarea sumamente tediosa, que consume gran cantidad de tiempo que el especialista puede dedicar al análisis y, además, propensa a cometer errores fruto de la envergadura de la tarea a realizar.

Condor es un middleware muy versátil para la ejecución de trabajos por lotes propia de la Computación de Alta Disponibilidad (High Throughput Computing, HTC) que en este caso permite llevar a cabo la ejecución concurrente de los diferentes procesos necesarios para obtener el estudio paramétrico.

Para complementar las capacidades de procesamiento y administración de tareas que dispone Condor es necesario diseñar una herramienta que administre y gestione la variación de parámetros, la generación de los diferentes juegos de datos, el procesamiento de los mismos y finalmente el postproceso de los resultados obtenidos.

En el contexto Método de Elementos Finitos (MEF) resulta conocido que todo análisis basado en el método conlleva las etapas de preproceso para generar los datos, el proceso o ejecución propiamente dicha y el postproceso de los resultados obtenidos. En este sentido Condor permite la ejecución concurrente de los diferentes procesos durante la etapa de ejecución con la consecuente disminución de los tiempos de procesamiento. Para ello es necesario contar con una infraestructura de computación distribuida adecuada como puede ser el caso de un cluster dedicado o de laboratorios de enseñanza que poseen tiempos ociosos y disponibilidad parcial.

La aplicación resultante, compuesta por la herramienta de administración, el código SOGDE y el middleware Condor, conforman una herramienta de cálculo paralelo automatizado que presenta particularidades frente a un código paralelo clásico basado en biblioteca como Message Passing Interface (MPI) por ejemplo. En este caso no es necesario modificar el código fuente que permanece exactamente igual al caso serial. La herramienta de administración propuesta complementa las capacidades de Condor y automatiza el flujo de tareas y datos necesarios para llevar a cabo el estudio paramétrico.

En la sección 2 del trabajo se presentan brevemente distintos modelos de Computación Distribuida con el fin de enmarcar Condor en el contexto de estas técnicas y proveer material de referencia. Luego en la sección 3 se discute Condor y se presentan sus características principales. El código SOGDE se comenta muy brevemente en la sección 4, antes de discutir la Arquitectura de la aplicación propuesta en la sección 5. Finalmente en la sección 6 se emplea la aplicación propuesta con el fin de generar en forma automática la construcción numérica de la hipérbola de Euler de una columna bidimensional (2D), que compara muy bien con los resultados teóricos disponibles.

Es importante destacar que la aplicación propuesta puede emplearse en diferentes problemas que requieren el procesamiento paramétrico y que el código de cálculo, en este caso SOGDE, puede reemplazarse por cualquier otro que resulte adecuado.

2 MODELOS DE COMPUTACIÓN DISTRIBUIDA

En el contexto de Computación Distribuida se pueden distinguir los siguientes paradigmas:

• Grid Computing: Este paradigma propone el acceso de recursos deslocalizados de diferente tipo: cómputo, almacenamiento, instrumentos, etc. El mismo se basa en la conformación de Organización Virtuales que comparten recursos. El nombre proviene de una analogía con la Red de Energía Eléctrica (Power Grid) ya que esta tecnología plantea el acceso a recursos computacionales en forma similar a la energía obtenida enchufando un equipo a la red.

• High Performance Computing (HPC): En este paradigma se prima la ejecución lo más rápido posible de las tareas. Conceptos tales como paralelización y multiproceso entran dentro de este ámbito, y su aplicación directa es hacer que cálculos que pueden durar semanas en un solo equipo se repartan entre varios, dividiendo el trabajo a realizar.

• High Throughput Computing (HTC): En este paradigma se prima la ejecución de la mayor cantidad posible de tareas. Conceptos como gestión de colas y de recursos son parte de este ámbito, y su aplicación directa pasa por la realización de la mayor cantidad de trabajos a lo largo del tiempo.

La Tecnología Grid Computing incluye como casos particulares a la Computación de Alto Rendimiento (HPC) y a la de Alta Disponibilidad (HTC). Mientras el primer paradigma es bastante conocido en nuestro país existen menos antecedentes para el caso de HTC.

El paradigma de HTC puede utilizarse con facilidad cuando se tiene una multitud de cálculos de tamaño mediano / pequeño, y se quiere agilizar la gestión de todos ellos.

3 ENTORNO DISTRIBUIDO CONDOR

Condor (Thain, Tannenbaum y Livny, 2002; Livny, Basney, Raman y Tannenbaum, 1997) es un entorno distribuido diseñado para Computación de Alta Disponibilidad (High Throughput Computing) y empleo de recursos ociosos. Este permite de manera transparente y simultánea explotar las capacidades de estaciones de trabajo distribuidas en un entorno LAN y/o campus y que pueden pertenecer a distintos individuos, grupos, departamentos e instituciones. La idea principal de Condor es hacer posible que cálculos con alto costo de gestión puedan ser realizados de forma ágil y eficiente.

3.1 Características de Condor

Como se puede observar en la Figura 1, la arquitectura del entorno Condor corresponde a un sistema distribuido de tipo centralizado. Una característica de este tipo de sistemas es que toda la gestión general del mismo esta a cargo de un único nodo maestro, mientras que los restantes nodos pertenecientes al sistema no establecen ningún tipo de comunicación entre ellos.

El conjunto compuesto por el nodo maestro y los demás nodos que intervienen en el sistema se lo conoce como pool de Condor. El funcionamiento de cada nodo que interviene en el pool dependerá de su configuración. Una configuración común es contar con nodos con capacidad para lanzar y recibir trabajos para su ejecución. Aunque en algunos casos resulta de utilidad contar con nodos que solamente permiten lanzar trabajos al sistema.

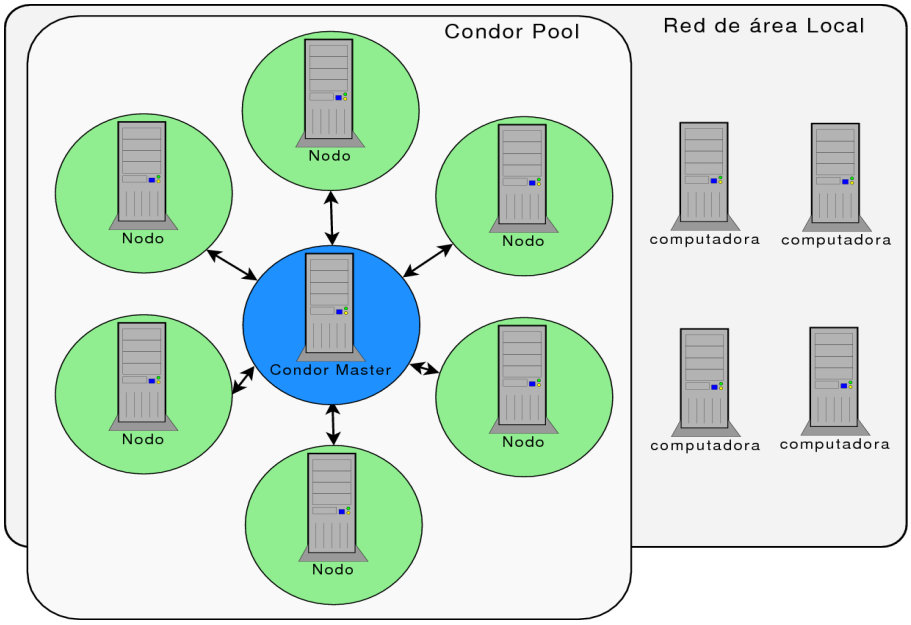

Figura 1: Arquitectura centralizada del entorno Condor.

3.2 Procesos Disponibles para Condor

Condor provee facilidades para ejecutar en forma distribuida diferentes tipos de problemas, para ello se vale de lo que en la jerga de Condor se conoce como universo. Condor posee diferentes tipos de universos que le permiten adaptar su comportamiento para resolver diferentes clases de problemas. Una lista completa de los universos de Condor puede consultarse en el manual del middleware (Condor Team, 2006).

Los universos estándar y vanilla son dos de los más comunes y simples de utilizar que provee Condor. Estos universos resultan adecuados para la ejecución de aplicaciones concurrentes de numerosos procesos independientes. Por ejemplo puede citarse el caso de las validaciones necesarias en problemas de Algoritmos Genéticos (Catania and García Garino, 2007; Catania y García Garino; 2008, Martínez et al, 2007), en donde resulta necesario repetir un mismo experimento un número adecuado de veces con el fin de determinar si se han obtenido resultados similares. Estos universos resultan adecuados para problemas en los cuales resulta necesario ejecutar la misma aplicación variando paramétricamente los datos de cada ejecución, técnica que se conoce como barrido de parámetros.

Una de las ventajas más importantes que surgen de la ejecución de este tipo de experimentos sobre Condor, consiste en la disminución de los tiempos de ejecución. Sin embargo, Condor no puede realizar la tarea por si solo, si no que se necesita de una aplicación que lleve a cabo las tareas relacionadas con la gestión de procesos y datos.

3.3 Ejecución de Trabajos

La manera más común de ejecutar un trabajo sobre el entorno Condor, consiste en utilizar los programas de línea de comandos provistos por el middleware.

Para poder ejecutar un trabajo dentro del entorno Condor, se necesita un archivo de texto en donde se describen las características del trabajo utilizando una sintaxis determinada.

Condor ofrece un gran número de opciones para describir las características y el comportamiento del trabajo que se desea ejecutar, se comentan aquí las más relevantes a los fines de este trabajo.

En la Figura 2 se muestra un ejemplo sencillo de un archivo para lanzar trabajos en Condor. En la línea 2 se indica el nombre del archivo que se desea ejecutar en Condor. Las líneas 4 y 5 permiten configurar nombres de archivos para redirigir la salida estándar y la salida de error estándar respectivamente. Mientras que en la línea 6 se indica el nombre del archivo donde Condor escribirá información relativa al estado del trabajo. Las líneas 7 y 8 indican que se debe realizar una transferencia del archivo ejecutable a la máquina remota. En casos donde Condor ha sido instalado sobre maquinas con sistemas de archivos compartidos (por ejemplo NFS) esta opción no resulta necesaria. En la línea 3 la opción Requirements permite definir los requerimientos que deben cumplir las maquinas remotas para poder ejecutar el trabajo. Para el caso mostrado en el ejemplo de la figura 2 se indica que las maquinas remotas deben ser máquinas con arquitectura de tipo INTEL de 32 bits. Finalmente en la línea 9 se indica el número de veces que el trabajo debe ser ejecutado sobre Condor. Para este caso el trabajo va a ser ejecutando una única vez.

```
1. Universe = vanilla
2. Executable = sogde
3. Requirements = (Arch == INTEL)
4. Output = sogde.out.$(Process)
5. Error = sogde.err.$(Process)
6. Log = sogde.log.$(Process)
7. Should_transfer_files = YES
8. When_to_transfer_output = ON_EXIT
9. queue 1
```

Figura 2: Archivo que describe un trabajo de Condor.

Para ejecutar el trabajo en el pool de Condor se utiliza el programa condor_submit, el cual recibe como parámetro el nombre del archivo que describe el trabajo a ejecutar. A su vez se puede monitorizar el estado del trabajo mediante otros programas provistos por Condor como condor_q y condor_status.

Una vez que el trabajo ha sido enviado para su ejecución en Condor, el middleware examina la disponibilidad de los recursos existentes en el sistema. Si se cuenta con equipos suficientes para cubrir los requerimientos, los trabajos son ordenados y enviados directamente. En caso de no haber equipos suficientes, se los ordena según prioridades y se los va ejecutando a medida que se encuentran equipos disponibles.

La gestión de recursos se realiza mediante unas etiquetas asignables tanto a los trabajos como a los recursos, que se denominan ClassAd y especifican las características de unos y otros. Una vez realizada esta caracterización, la gestión de recursos únicamente consiste en emparejar los ClassAd de trabajos con los ClassAd de recursos.

Una vez que Condor encuentra ejecutar un trabajo, puede transferir tanto el programa como los ficheros necesarios para su ejecución al recurso adecuado y comenzar con la ejecución. Al finalizar la ejecución los resultados se transfieren de nuevo al equipo del usuario.

Entre las características más interesantes que ofrece Condor, esta la posibilidad de permitir la ejecución de los programas sobre Grid Computing. De esta manera es posible extender los recursos computaciones y emplear la potencia de cálculo de otras maquinas que se encuentren fuera de la red local. Condor provee mecanismos que le permiten conectarse de manera directa con Globus Toolkit (Globus Toolkit, 2008) mediante Condor-G y el soporte del sistema de seguridad GSI (Thain, Tannenbaum y Livny, 2003).

4 CÓDIGO SOGDE

El código de elementos finitos denominado SOGDE, el cual fue desarrollado por García Garino (García Garino, 1993; García Garino y Oliver, 1995; García Garino y Oliver, 1996), y emplea una formulación constitutiva basada en hiperelasticidad y en cinemática multiplicativa del tensor gradiente de deformación, pudiéndose encontrar una amplia descripción de la misma en las referencias antes mencionadas. Como caso particular posee la capacidad de encontrar trayectorias de equilibrio no lineales.

El código SOGDE ha sido ampliamente validado en la solución de problemas no lineales. En las referencias (García Garino, 1993; García Garino y Oliver, 1995; García Garino y Oliver, 1996; García Garino, Gabaldón y Goicolea, 2006) se discuten problemas que muestran comportamiento postpico con ablandamiento y endurecimiento. Además en trabajos recientes de algunos de los autores se tratan problemas de pandeo tridimensional (Careglio, Mirasso y García Garino, 2006; Careglio, Mirasso y García Garino, 2007) similares al propuesto en este caso.

Asimismo para facilitar el postproceso gráfico de los resultados se ha incluido una interfaz con el software GID (Ribó, 2006).

5 ARQUITECTURA DE LA APLICACIÓN

Como se señaló en la sección 3.2 es necesario diseñar una herramienta de administración que explote las posibilidades de Condor para resolver trabajos por lotes en forma concurrente. El lector familiarizado con el empleo de códigos de Elementos Finitos reconocerá que en todo análisis se deben llevar a cabo tareas de preproceso, antes de la ejecución propiamente dicha del código del Método de Elementos Finitos (MEF) y el postproceso de los resultados, como se indica en la Figura 3.

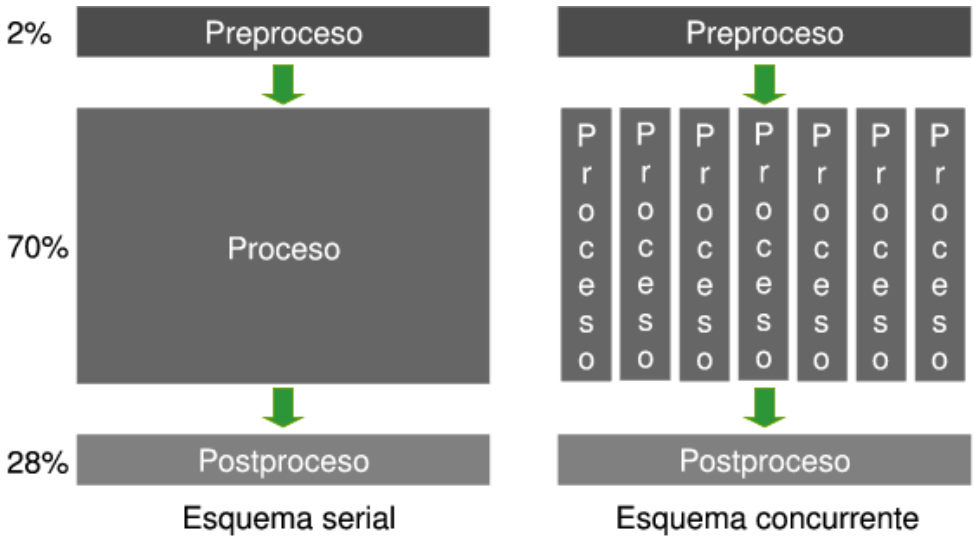

Figura 3: Esquemas serial y concurrente del proceso de un estudio paramétrico.

En la Figura 4 se muestra un modelo conceptual de la arquitectura de la aplicación, la cual se ha escrito en Python, ya que permite desarrollar prototipos de producción en corto plazo. En la misma se distinguen las etapas de preproceso, ejecución y postproceso señaladas en el párrafo anterior. Es importante señalar que el Master del pool se encarga de llevar a cabo todas las tareas de pre y postproceso, mientras que la ejecución propiamente dicha se ejecuta de manera concurrente.

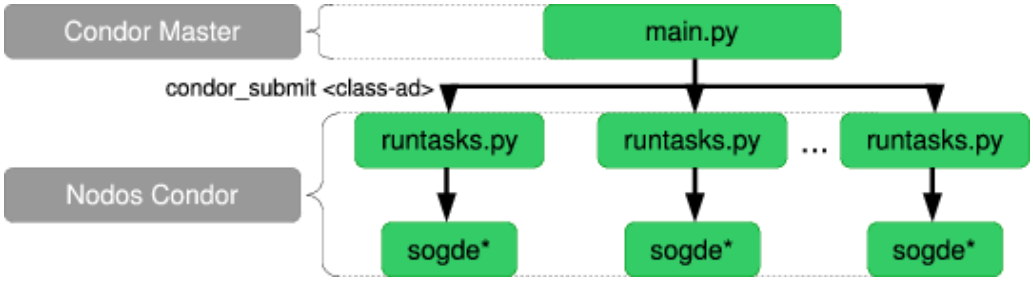

Figura 4: Arquitectura de la aplicación.

```
import sys

from processor.processor import Processor
from processor.processorException import ProcessorException
from calculations.criticalChargeCalculator import CriticalChargeCalculator

#functions
def errorExit(message):
    print "\nerror:: " + message print "finishing with errors"
    sys.exit(0)

#main
if __name__ == "__main__":
    #loading of the configuration file
    try:
        module = sys.argv[1][0:-3]
        exec "from " + module + " import *"

        print "> config file '" + module + "' found."
    except:
        errorExit("Config file missing using")

    #instantiation of the critical charge calculator
    criticalChargeCalculator = CriticalChargeCalculator(b, h, t, e, k)

    try:
        #instantiation of the processor
        processor = Processor(dataFileName, nodesFileName, scalesFileName,
namePrefix, outDirectory, sogdePath, comparingNode,
criticalChargeCalculator, originalL)
        processor.setDeleteTemporaryFiles(deleteTemporaryFiles)
        processor.setParallelProcessing(parallelProcessing)
        processor.setPackageQuantity(packageQuantity)

    except NameError, e:
        errorExit("Configuration property not defined: " + str(e))

    #execution of the processor
    try:
        processor.start()
        except ProcessorException, e:
        errorExit("Processor Exception: " + e.message)
```

Figura 5: Script main.py.

```
import ...

class Processor(threading.Thread):

    def __init__(self, dataFileName, nodesFileName, scalesFileName,
namePrefix, outDirectory, sogdePath, comparingNode,
criticalChargeCalculator, originalL):

    def computeTiming(self, function):
        timer = Timer(True)
        function()
        timer.stop()

        timing = timer.getTiming("ms")
        self.appendTiming(timing)

        print "timing: " + str("%.3f" % timing) + "ms"
    ...
    #body
    def run(self):
        self.configure()
        self.computeTiming(self.generateInputFiles)
        self.computeTiming(self.runSogde)
        self.computeTiming(self.generatePlotFiles)
        self.computeTiming(self.plot)
        if self.deleteTemporaryFiles:
            self.computeTiming(self.cleanFiles)
        self.showTimings()
    ...
```

Figura 6: Clase Processor (processor.py).

El programa se inicia mediante la ejecución del archivo main.py (ver Figura 5), el cual contiene la lógica principal de la aplicación, y se encarga de pasar al proceso controlador los datos de configuración necesarios para poder efectuar el barrido de parámetros (Ver Figura 6). Estos datos se encuentran en un archivo de configuración con sintaxis de python, lo que permite realizar una importación directa de dichos datos al programa.

Una vez cargados, los datos de configuración, comienza la etapa de preprocesamiento, en donde a partir del archivo de configuración se generan distintos archivos de datos (que se discuten en la sección 5.1).

El proceso de ejecución de SOGDE en las maquinas planificadas por Condor se lleva a cabo mediante el script runtasks.py.

Finalmente una vez que se han procesado todos los casos de estudio, comienza la etapa de post procesamiento en donde se procede a la extracción de datos para la generación archivos de resultados y gráficas. Se genera un archivo de log de todo el proceso, el cual contiene: i) datos del archivo de configuración especificado; ii) cantidad de casos de estudio a evaluar según cantidad de nodos y escalas y iii) evaluación de tiempos según las distintas tareas en segundos y también en porcentaje respecto del total.

A continuación se describen con mayor detalle como se han implementado cada una de las tres etapas citadas dentro de la aplicación de gestión.

5.1 Pre-procesamiento

Dado que en el estudio de problemas paramétricos se deben resolver numerosas aplicaciones con diferentes datos, en esta etapa se debe generar el conjunto de archivos para

SOGDE, adecuado al problema de interés. Una aproximación sencilla sería generar en forma independiente cada archivo con un programa adecuado como GID, GMesh, o similar. Sin embargo resulta mucho más eficiente generar dichos archivos de forma automática variando los parámetros de interés de manera automática mediante una aplicación diseñada a tal fin. Para ello se cambian convenientemente coordenadas del modelo, con el fin de variar esbelteces y también se cambian la intensidad de la carga y su punto de aplicación con el fin de variar imperfecciones.

5.2 Ejecución Distribuida

La etapa de procesamiento correspondiente al modo de ejecución concurrente comienza con el armado de paquetes. Cada uno de ellos corresponde a un slice y es un archivo comprimido (tar.gz); los cuales están nombrados como sogde-package-xxxxxx.tar.gz, donde xxxxxx es un número del slice al que corresponde el paquete.

Los paquetes contienen:

• un conjunto de archivos de datos (un subconjunto del total generado).

• el solver de elementos finitos (SOGDE).

• un archivo de parámetros para la ejecución de los procesos correspondiente para dicho paquete (slicexxxxxxData.py). Este archivo es un módulo de python cargado dinámicamente por el script runtasks.py en el nodo planificado.

La construcción de los paquetes se lleva a cabo con el fin de disminuir los tiempos de comunicaciones, como se mostrará en la sección 6.4. La compresión de archivos de texto con diferencias mínimas entre ellos resulta en una disminución importante en el volumen de datos a ser transferidos por la red.

También brinda un manejo mas ordenado de los archivos (que dependiendo de la configuración podrían llegar a ser muchos). Como consecuencia del empaquetado se obtiene un archivo de submit de trabajos mucho más legible.

Con la generación de paquetes concluida se procede además al armado del archivo de submit; ClassAd de Condor. El cual se encuentra dividido en dos secciones como se observa en la Figura 7.

• La sección general contiene información común de configuración para todos los trabajos a submitir. En la misma se especifica: el nombre del ejecutable (en nuestro caso es el script runtasks.py presentado en la Figura 8), el universo de ejecución (vanilla), el directorio inicial desde el cual Condor busca los archivos de entrada y en el cual depositará todos los archivos generados por cada uno de los jobs, el archivo de log para los trabajos y, finalmente las instrucciones para transferir los archivos de salida, lo cual se realiza al terminar cada proceso.

• La sección de los slices contiene información particular de configuración para cada uno de los trabajos: los argumentos de ejecución para el script del trabajo (el nombre del paquete y el nombre del archivo de parámetros para el trabajo) y el archivo a ser transferido al momento de ejecutar el trabajo (el paquete).

```
# --- general section ---
Executable = runtasks.py
Universe = vanilla
Initialdir = tmp/
Log = sogde-20080630204024.log
Should_transfer_files = YES
When_to_transfer_output = ON_EXIT

# --- slices section ---
# <slice 000000>
Arguments = sogde-package-000000.tar.gz slice000000Data.py
Transfer_input_files = sogde-package-000000.tar.gz
Queue

...

# <slice 000011>
Arguments = sogde-package-000011.tar.gz slice000011Data.py
Transfer_input_files = sogde-package-000011.tar.gz

Queue
```

Figura 7: Ejemplo de ClassAd generado.

Una vez generado el ClassAd se realiza la invocación del comando condor_submit <class-ad> con el cual Condor toma el control del proceso y se le delega la administración y ejecución de los trabajos. La espera de finalización de los trabajos en el programa se realiza mediante la invocación del comando condor_wait <log-file>, este comando no devuelve el control de programa hasta que la ejecución de todos los trabajos finalice. La espera se hace sobre el archivo de log especificado en el ClassAd, cuyo nombre debe ser pasado como parámetro a dicho comando.

El archivo de datos adicional slicexxxxxxData.py permite organizar la ejecución de cada job manteniendo inalterado el script runtasks.py. El desacople de los datos en un archivo separado permite la existencia de un único script de ejecución independiente de los datos el cual es replicado para cada unos de los distintos trabajos a realizar. El script comienza por la descompresión del paquete, posteriormente importa dinámicamente el archivo de datos adicionales que se encontraba dentro del paquete. Una vez que los datos fueron importados comienza la ejecución del las distintas instancias del solver indicando el archivo de datos y el archivo de salida correspondiente para cada una de los casos de estudio. Concluido el calculo de los distintos casos de estudio correspondientes, Condor se encarga de la transferencia de los archivos de salida, al directorio inicial especificado en el ClassAd (Initialdir = tmp/) de la máquina desde la cual se submitieron los trabajos. Dichos archivos quedan entonces a disposición del script principal para proseguir con la etapa de post-procesamiento una vez que todos los trabajos sean concluidos.

5.3 Post-procesamiento

Con la ejecución de los cálculos de todos los casos de estudio se tienen a disposición todos los archivos de salida de cada una de las ejecuciones del solver, estos son analizados para obtener resultados de interés que permitirán generar diferentes curvas. Además se generan archivos que contienen los comandos necesarios para generar estas gráficas.

Las curvas generadas se tratan en detalle en la sección siguiente.

```
#!/usr/bin/python

import ...

def execute(command):
    #print " > " + command
    os.system(command)
    if __name__ == "__main__":
        __startTime = time.time()

        #get package
        packageName = sys.argv[1]
        print " :: using package " + packageName

    #deploy package
    command = "tar -xzf " + packageName
    execute(command)

    #import datafiles registry
    moduleFile = sys.argv[2]
    module = moduleFile[0:-3]
    try:
        exec "from " + module + " import *"
    except:
        execute('echo "finished with errors" > abort-tasks.sync')
        raise "datafiles config not found: " + module

    """ variables in the module:
        - sliceName
        - dataFiles[]
        - outFiles[]
        - localSogdePath
    """

    execute("chmod +x " + localSogdePath)

    #running sogde
    for i in range(len(dataFiles)):
        dataFile = dataFiles[i]
        outFile = outFiles[i]
        exeLine = "./" + localSogdePath + " <" + dataFile + " >" +
outFile
        execute(exeLine)

        print "   > condor process for package '" + packageName + "'
terminated."

    #cleaning files
    for detaFile in dataFiles:
        os.system("rm " + dataFile)

    os.system("rm " + localSogdePath)
    os.system("rm " + moduleFile)
    os.system("rm " + moduleFile + "c")

    __endTime = time.time()
    timeString = "time: " + str(__endTime - __startTime) + "s, start:" +
str(__startTime) + "s, end:" + str(__endTime) + "s"
    execute('echo "' + timeString + '"')
```

Figura 8: Script runtasks.py.

6 ESTUDIO PARAMÉTRICO DE LA COLUMNA DE EULER 2D

Con el fin de ilustrar las posibilidades de la aplicación propuesta, se presenta la ejecución automática de un estudio paramétrico. Mediante dicho estudio se obtiene la llamada hipérbola de Euler del problema de estabilidad del equilibrio de columnas, ampliamente tratado en la bibliografía (Bleich, 1952; Timoshenko, 1982). Para alcanzar este objetivo se deben resolver alrededor de 200 problemas no lineales con SOGDE.

En la sección 6.1, define el problema y los posibles resultados a obtener con SOGDE. En la sección 6.2 se especifican la variación de los parámetros de interés y finalmente en la sección 6.3 se discuten los resultados del estudio paramétrico.

6.1 Definición y análisis del problema

En mecánica estructural uno de los modos de falla a considerar es la pérdida de estabilidad del equilibrio de columnas esbeltas, la cuales son sensibles a perder estabilidad debido a la aplicación de una carga de compresión en la misma.

En la Figura 9 se muestra esta tipología estructural, para la situación perfecta junto a la situación imperfecta que se considera en este trabajo. Son columnas de longitud *L*, que se encuentran empotradas en un extremo y libre en el otro, y se le aplica una carga axial *P* de compresión en el extremo libre de la misma con excentricidades *e* respecto del eje central.

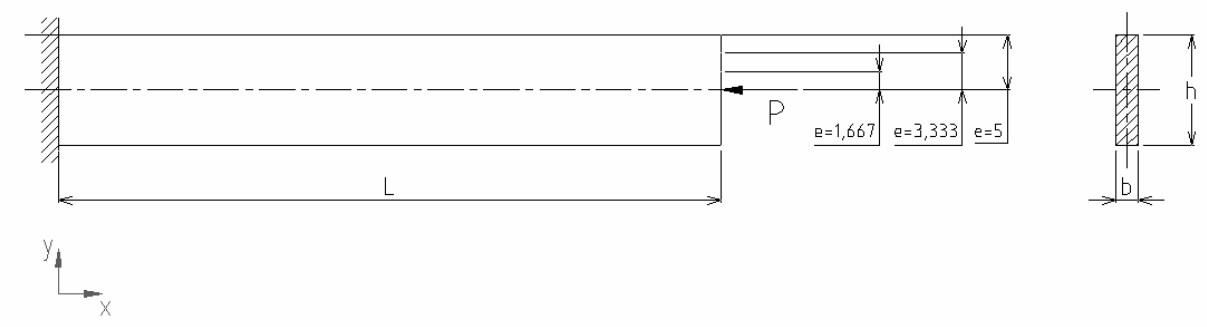

Figura 9: Geometría de la columna perfecta, vínculos y carga aplicada.

Cuando se analiza la estructura perfecta el valor de carga para el cual aparece una bifurcación del equilibrio de la trayectoria fundamental lineal, se denomina carga crítica y está dado por (Timoshenko, 1982):

$$P_{cr} = \frac{\pi^2 EJ}{4L^2} \tag{1}$$

donde *E* es un parámetro del material conocido como módulo de Young y $J=bh^3/12$ el momento de inercia el cual depende de la sección de la columna. Para ese valor de carga crítica además de la trayectoria fundamental contenida en el eje longitudinal de la carga, existen configuraciones de equilibrio que tienen desplazamientos transversales no nulos.

Es posible analizar el comportamiento de estructuras perfectas mediante las trayectorias de equilibrio no lineales (como las mostradas en la Figura 10) de estructuras imperfectas asociadas. Estas últimas son obtenidas a partir de la estructura perfecta a la que se le incorpora algún tipo de imperfección, como lo son las excentricidades planteadas en la Figura 9.

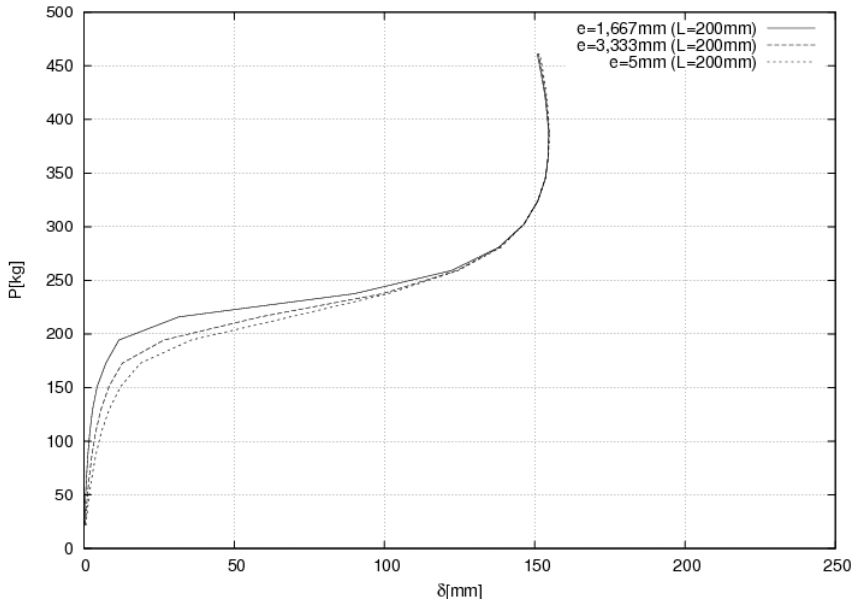

Figura 10: Trayectorias de equilibrio no lineales para las tres imperfecciones estudiadas con L=200mm.

Para un determinado valor de L, por ejemplo 200 mm, cuando se considera el caso de existencia de imperfecciones, como lo son las excentricidades planteadas en la Figura 9, la respuesta resulta como las trayectorias de equilibrio no lineales mostradas en la Figura 10, obtenidas con el código SOGDE citado en la sección 3 (Careglio, Mirasso y García Garino, 2006). Estas trayectorias se encuentran representadas por la evolución de los valores de carga aplicada P (kg) en función del desplazamiento transversal del extremo de la ménsula δ(mm). Para niveles elevados de la carga P aplicada, superiores al valor P_{cr}, la columna adopta una configuración pandeada como la mostrada en la Figura 11.

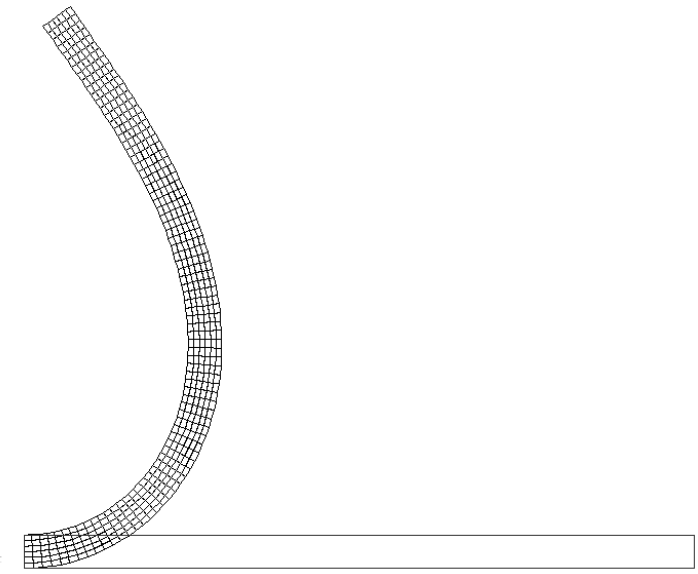

Figura 11: Configuración original y deformada.

A partir de estas trayectorias de equilibrio no lineales es posible inferir en forma numérica el valor de la carga crítica de la estructura sin imperfección lo cual se realiza utilizando gráficos de Southwell (Croll y Walker, 1975; Timoshenko y Gere, 1985). En la Figura 12 se

muestran dichos gráficos asociados a cada imperfección. El valor de la carga crítica de la estructura perfecta es determinado a partir del valor de la pendiente de cada recta asociada a la correspondiente longitud, siendo el valor de esta pendiente igual para las distintas imperfecciones asociadas a una misma longitud. En este trabajo en particular se obtienen las rectas correspondientes a partir de los dos primeros incrementos de cargas.

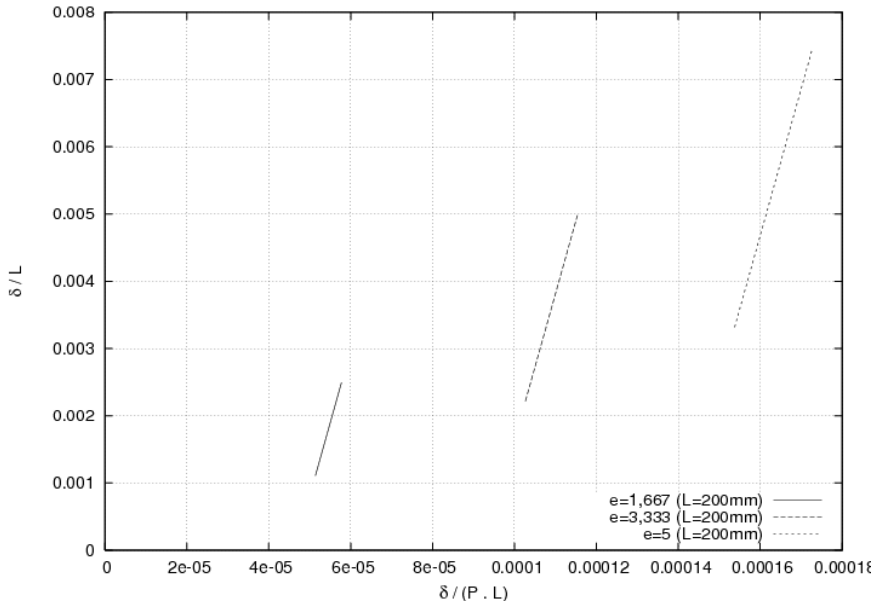

Figura 12: Gráficos de Southwell, L=200mm, para las tres imperfecciones.

Una vez que se ha determinado numéricamente el valor de la carga crítica perfecta se puede dividir a la misma por el área de la sección transversal de la columna, lo cual permite obtener un valor de tensión crítica σ_{cr} asociada a una determinada longitud. En la Figura 13 se muestra el valor de tensión σ_{cr} en función de la esbeltez de la columna definida como $\lambda=2L/r$ para un radio de giro r=2,887mm y L=200mm, lo cual puede compararse con el valor teórico que resulta de dividir la ecuación (1) por el área en cuestión. Si este valor teórico se calcula para distintas longitudes, Figura 13, se obtiene la conocida Hipérbola de Euler (Bleich, 1952; Timoshenko, 1982).

Para obtener otros puntos de la hipérbola de Euler es necesario variar la longitud de la columna L, manteniendo su sección transversal constante.

Por último es de interés estudiar la sensibilidad a las imperfecciones de este tipo de estructura, esto puede realizarse analizando la rigidez a flexión de la columna. Para ello se puede evaluar cómo cambia la rigidez con relación a la rigidez inicial E_0, en cada trayectoria de la Figura 10, a medida que aumenta la carga aplicada (Careglio, Mirasso y García Garino, 2007). De esta manera es posible aproximar la rigidez tangente con una rigidez secante incremental E_S definida como:

$$E_S = \left(P_{i+1} - P_i\right) \Big/ \left(\delta_{i+1} - \delta_i\right) \tag{2}$$

Esto último es representado en forma gráfica por la Figura 14 en la cual aparece la relación E_s/E_o en función de la carga aplicada P para cada una de las imperfecciones para la longitud analizada. De esta forma puede apreciarse que a medida que aumenta la carga disminuye la rigidez de la columna.

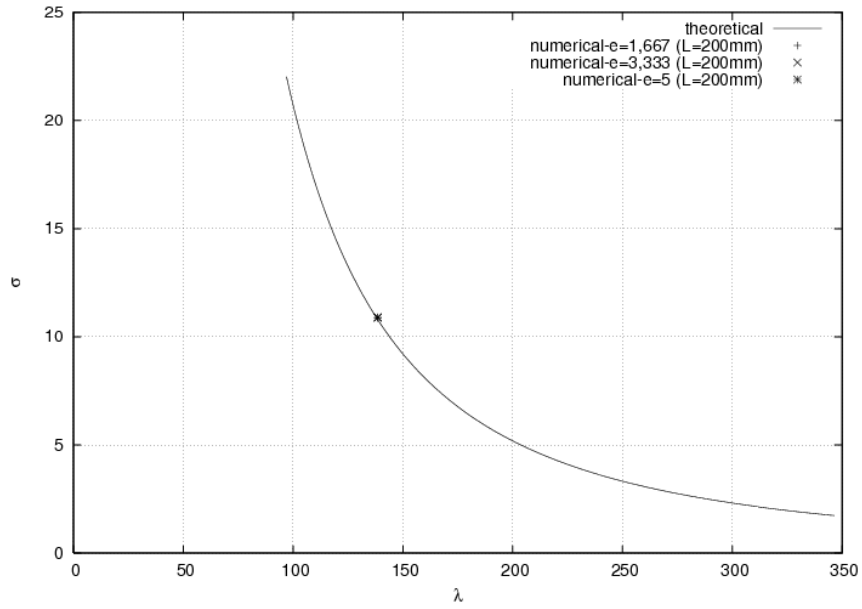

Figura 13: Hipérbola de Euler obtenida en forma teórica y σ_{cr} para L=200mm obtenida en forma numérica.

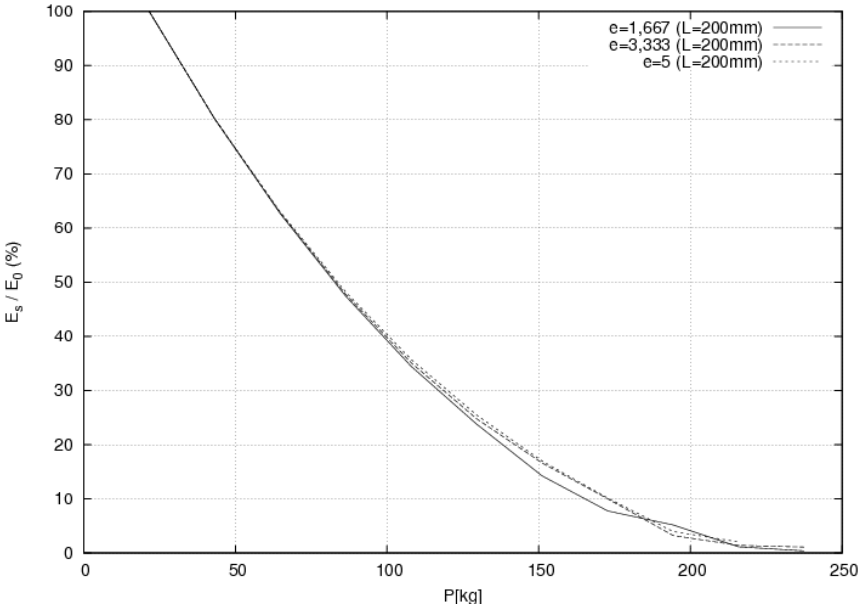

Figura 14: Sensibilidad del cambio de rigidez a la imperfección para L=200mm.

Además es necesario aclarar que para una determinada longitud y para cada imperfección sería necesario generar en forma manual un archivo de datos obteniéndose un archivo de resultados el cual debe procesarse para obtener cada una de las curvas. Esto es presentado en forma esquemática en la Figura 15, donde también se muestra a modo de resumen el orden seguido para obtener las distintas curvas de interés.

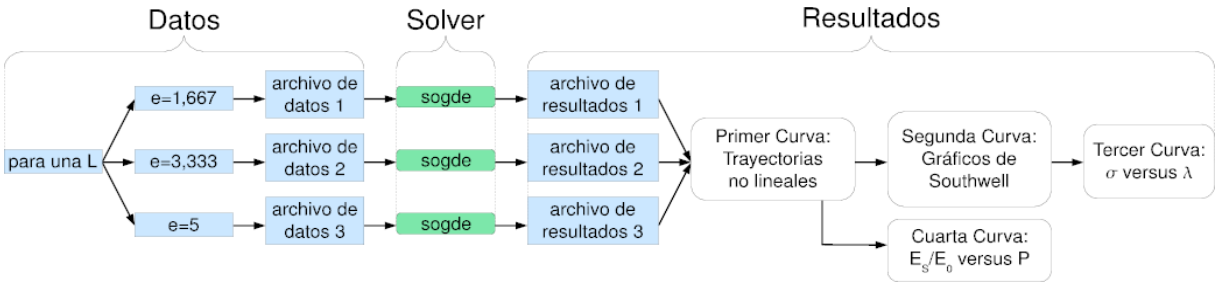

Figura 15: Secuencia en la generación de los gráficos de interés.

6.2 Definición del estudio paramétrico

Con el fin de estudiar la estructura para distintas longitudes L e imperfecciones e, se lleva a cabo un estudio paramétrico. Con este fin se varían las longitudes L y las excentricidades e definidas en la Figura 9.

Se han analizado longitudes L que van desde L=140mm hasta L=500mm. Se ha adoptado una sección transversal uniforme de forma rectangular con una altura h=10mm y un espesor b=2mm, que se mantiene invariante en todos los casos. Con respecto a las imperfecciones, se han adoptado excentricidades con valores de e= 1,667mm, 3,333mm y 5mm.

El material empleado es un material elástico lineal con E=21000 kg/mm^2 y ν=0,3. Se ha discretizado la geometría antes mencionada con 80 elementos en la longitud y 6 en la altura, empleándose en todos los casos el elemento cuadrilátero Q1 estándar (Zienkiewicz y Taylor, 1991).

El cálculo de la tensión σ_{cr} para los diferentes casos procesados permite así reproducir la hipérbola de Euler del problema.

6.3 Discusión de resultados del estudio paramétrico

En la primera gráfica obtenida se presentan las trayectorias de equilibrio no lineales para las distintas excentricidades en forma similar a la Figura 10 pero en este caso para todo el rango de longitudes estudiadas como se muestra en la Figura 16.

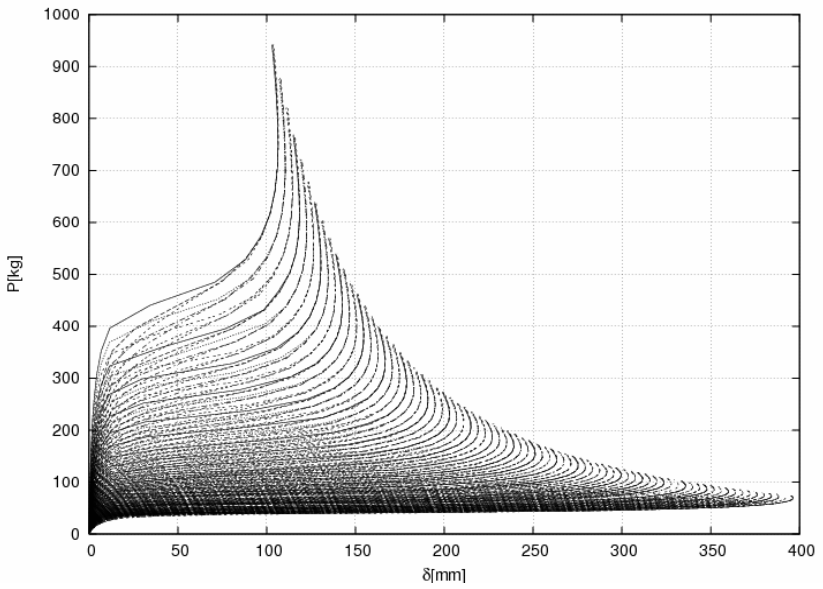

Figura 16: Trayectorias de equilibrio no lineales para las longitudes e imperfecciones estudiadas.

La curva superior corresponde a la menor longitud estudiada (L=140mm) para la menor imperfección, mientras que la curva inferior corresponde a la mayor longitud (L=500mm) para la mayor imperfección. En la misma figura puede observarse que se ha logrado alcanzar grandes desplazamientos verticales, siendo el mismo superior a 10 veces el valor de la altura de la sección para la curva superior y cercano a 40 veces para la curva inferior.

Luego a partir de las trayectorias de equilibrio se obtienen Gráficos de Southwell para cada longitud e imperfección lo que se muestra en la Figura 17. Puede apreciarse que se han obtenido tres grupos de curvas. El grupo superior de rectas corresponde a la mayor imperfección, el grupo del centro a la excentricidad intermedia y el grupo inferior a la menor imperfección. A su vez en cada uno de los tres grupos la primer recta de la izquierda corresponde a la menor longitud de la columna (L=140mm) y por lo tanto una mayor pendiente de la recta lo que corresponde a un mayor valor de carga crítica de la estructura perfecta; mientras que las rectas en el extremo derecho corresponden a la mayor longitud (L=500mm) teniendo dichas rectas menor pendiente y por lo tanto menor valor de carga crítica de la estructura perfecta. Las rectas intermedias en cada grupo corresponden a las longitudes intermedias entre la mayor y la menor longitud de la estructura.

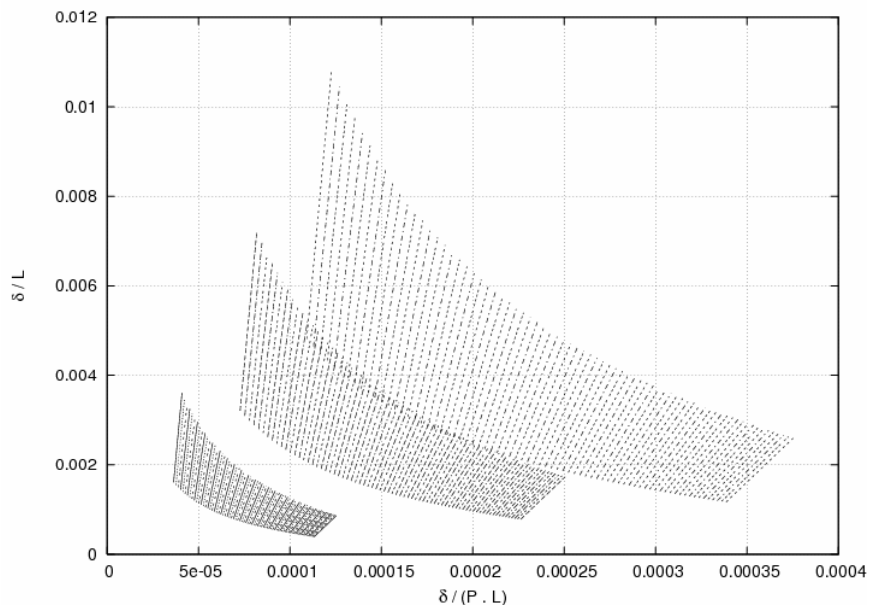

Figura 17: Gráficos de Southwell para cada longitud e imperfección.

Como se explicó en el punto 6.1 a partir de las curvas anteriores es posible obtener los valores de tensión crítica de la estructura perfecta y compararlos con los resultados teóricos. En la Figura 18 se muestran dichos valores numéricos representados por puntos junto a los valores teóricos representados por la línea continua para las longitudes estudiadas. El primer punto de la izquierda corresponde a la longitud inferior, el último de la derecha a la longitud superior y los puntos intermedios entre estos dos extremos a las longitudes intermedias.

En la mencionada figura puede apreciarse que existe una buena correlación entre los resultados numéricos obtenidos y los teóricos, en particular para los valores más bajos de esbeltez donde la determinación de la tensión crítica es prácticamente exacta. Para los mayores valores de esbeltez la diferencia entre las curvas aumenta, esto puede ser debido a la gran no linealidad presente en las trayectorias de equilibrio incluso cuando se obtienen a partir de incrementos de carga pequeños, con lo cual puede haber una perdida de exactitud en la determinación de la carga crítica a partir de los gráficos de Southwell. Cabe destacar que para

la mayor esbeltez estudiada existe una relación longitud/alto de la sección de 50 a 1 la cual es de poca aplicación práctica, pudiendo considerarse como aceptable la diferencia entre la tensión crítica calculada en forma numérica y teórica.

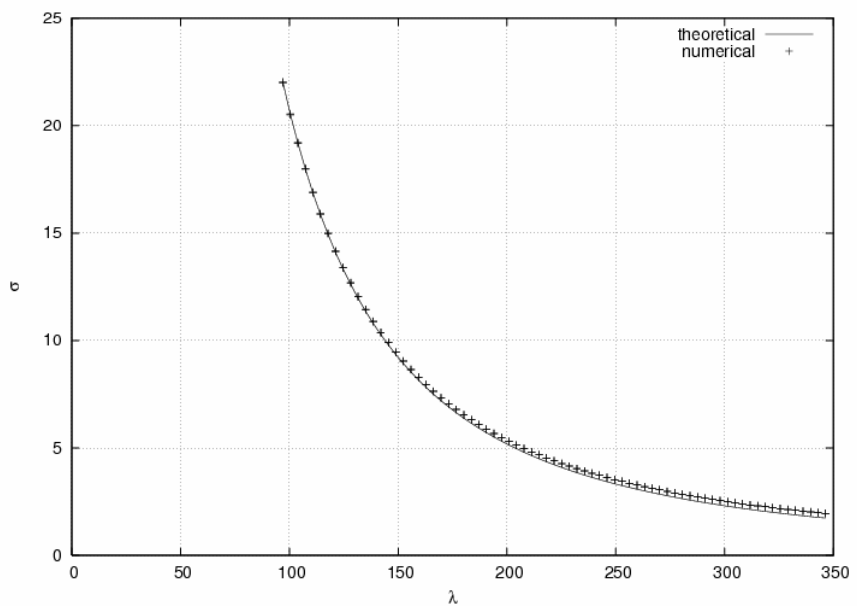

Figura 18: Hipérbola de Euler para las longitudes estudiadas.

Además de las curvas la herramienta desarrollada genera un informe con los resultados numéricos y teóricos obtenidos para las longitudes estudiadas, Tabla 1, en donde a fines ilustrativos, solo se han mostrado dos longitudes de todas las que han sido estudiadas. En la primera columna de la Tabla 1 se muestra el número de nodo de la malla de elementos finitos en donde se aplica la carga junto con el factor de escala por el cual se multiplica la longitud original. En la segunda aparece la esbeltez correspondiente, en la tercera y cuarta el valor de carga y tensión crítica respectivamente, calculada en forma numérica, y en la quinta columna el valor de la tensión crítica teórica que para este ejemplo conocida.

Nodo- Escala	Esbeltez	Pcr	Tensión	Tension teórica
22-1.40	96.994845	439.522674	21.976134	22.030367
33-1.40	96.994845	440.233644	22.011682	22.030367
43-1.40	96.994845	440.790237	22.039512	22.030367
22-1.45	100.458947	410.005761	20.500288	20.537227
33-1.45	100.458947	410.619775	20.530989	20.537227
43-1.45	100.458947	411.088176	20.554409	20.537227

Tabla 1: Informe con resultados numéricos y teóricos.

Por último se presenta en la Figura 19 la relación E_s/E_o versus la carga aplicada P para cada una de las imperfecciones (como en la Figura 14) pero para todas las longitudes analizadas. La primer curva a la izquierda corresponde a la mayor longitud (L=500mm) para la mayor imperfección, mientras que la última de la derecha corresponde a la menor a la menor longitud (L=140mm) para menor imperfección. Además se muestran los resultados obtenidos para las longitudes intermedias.

Dicha figura indica, como era de esperar, que en la columna de mayor longitud la pérdida de rigidez se produce a valores inferiores de carga que para longitudes más cortas. Además se observa que, para una longitud L dada, las curvas de E_s/E_o en función de P son muy similares para las tres imperfecciones estudiadas. Es decir, que si bien las trayectorias de equilibrios son distintas (Figura 16), los cambios de rigidez E_s/E_o para cada trayectoria de cada imperfección para una misma longitud de la columna son prácticamente iguales.

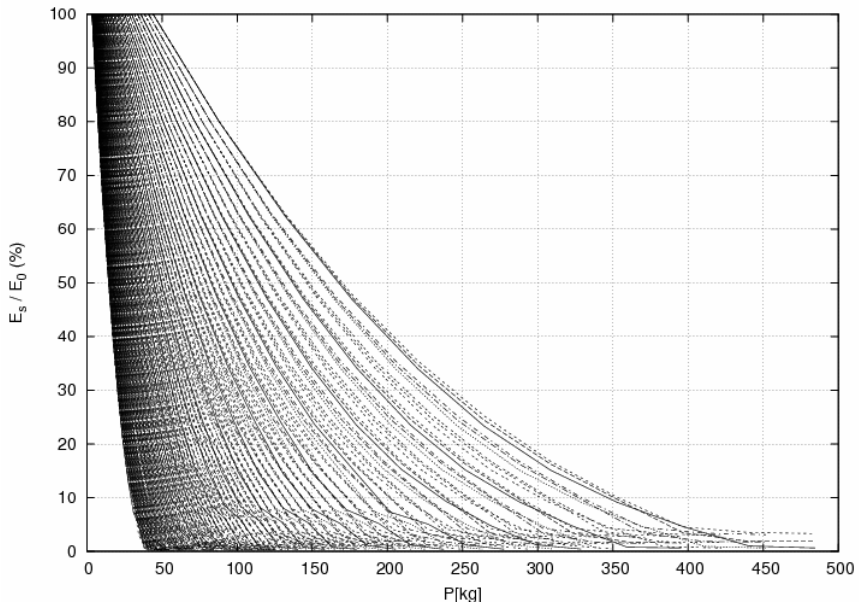

Figura 19: Sensibilidad del cambio de rigidez a la imperfección.

Se debe destacar que dado que los valores de cargas de inestabilidad elásticas dependen de la sección transversal de la columna surge la posibilidad, mediante la utilización de la herramienta desarrollada, de determinar las cargas de inestabilidad elástica de estructuras perfectas para distintos tipos de estructuras e imperfecciones que las empleadas en este trabajo.

En forma resumida puede decirse que para este estudio paramétrico los cambios de longitud y excentricidad se relacionan con lo expuesto en el punto 5.1, mientras que la generación de los gráficos y el informe se relacionan con el punto 5.3.

6.4 Métricas

Como fue expuesto en la sección 5.2, el empaquetado de archivos para su distribución en los nodos de Condor resulta ventajoso en cuanto a la disminución de los tiempos de comunicaciones. Esto puede verse en la siguiente evaluación.

Para 219 casos de estudio distintos distribuidos en 4 paquetes (aproximadamente 55 archivos de datos por paquete) tenemos que el volumen de información de cada paquete es de 968 KiB (991.546 bytes), mientras que el tamaño de todos los archivos en forma separada asciende a 2,96 MiB (3.111.215 bytes). Con lo que tenemos una disminución del volumen de datos de aproximadamente el 68,1% gracias al empaquetado. Este porcentaje varía de acuerdo a la cantidad de archivos de datos que sean incluidos en el paquete.

Otra de las ventajas del empaquetado es que se disminuye al mínimo el overhead generado por el protocolo para la preparación de la transmisión. Para realizar la transferencia de los archivos por separado dicho tiempo de preparación aparece para cada uno de los archivos a

transmitir, lo cual eleva el tiempo de comunicaciones. Con el empaquetado reducimos este tiempo ya que solo se produce para cada uno de los paquetes.

En la Tabla 2 se muestran para la ejecución concurrente de 219 casos de estudio en distinta cantidad de máquinas, los distintos tiempos. En la primer columna el lector puede ver la cantidad de maquinas utilizadas, en la segunda los tiempos de duración de la etapa de proceso medida en segundos, en la tercer columna el tiempo de duración total de la ejecución del programa. En la cuarta columna se puede apreciar el porcentaje de mejora haciendo una comparación entre los tiempos de duración de la etapa de proceso en forma serial y concurrente. En la quinta columna se puede apreciar el porcentaje de mejora de los tiempos globales de ejecución en forma serial y concurrente.

Los tiempos de demora de la ejecución serial son:

- 121 segundos en la etapa de proceso

- 177 segundos en la ejecución completa de la aplicación.

Máquinas	Proceso Conc.[s]	Total[s]	Mejora Proceso	Mejora Global
2	111	166	8,26%	6,21%
3	92	146	23,97%	17,51%
4	81	135	33,06%	23,73%
5	82	137	32,23%	22,60%
6	80	135	33,88%	23,73%
7	73	128	39,67%	27,68%
8	72	126	40,50%	28,81%
9	75	129	38,02%	27,12%
10	75	130	38,02%	26,55%

Tabla 2: Métricas para ejecuciones concurrentes utilizando distinta cantidad de máquinas.

De los resultados obtenidos surgen varios comentarios: el mejor tiempo de cálculo se alcanza para el caso de 8 máquinas. En general el tiempo total es la suma de tiempo de proceso más tiempo de comunicaciones. A medida que aumenta el número de equipos disponibles disminuye el tiempo de proceso, pero aumenta el tiempo de comunicaciones, y en general el mínimo se alcanza en un punto intermedio, en este caso 8 equipos. Cabe plantear la posibilidad de analizar con más detalle este aspecto para tomar ventajas de un número de equipos disponibles.

Otro punto importante para discutir es el procesamiento concurrente del postproceso, etapa que toma alrededor del 25-30% del estudio como se muestra en la Figura 3. De esta manera se conseguirá disminuir aun más el tiempo de procesamiento total.

7 CONCLUSIONES Y TRABAJOS FUTUROS

En el trabajo se ha presentado una herramienta capaz de realizar complejos estudios paramétricos de manera concurrente y automatizada.

Los resultados obtenidos con el caso de estudio analizado permiten concluir que los entornos de Computación de Alta Disponibilidad (HTC) en general y Condor en particular, conjuntamente con herramientas de Mecánica Computacional como es el caso de SOGDE conducen a importantes ahorros de tiempo de procesamiento y de análisis en el caso de

estudios paramétricos.

Las herramientas de HTC, como es el caso de Condor, bastante menos conocidos en nuestro país que las herramientas de HPC permiten potenciar con relativa simplicidad herramientas de Mecánica Computacional como resultan códigos de Elementos Finitos con capacidades no lineales como es el caso de SOGDE.

La hipérbola de Euler de una columna 2D perfecta, obtenida mediante la determinación de estados críticos de múltiples columnas imperfectas y con distintas esbelteces, tiene una muy buena precisión respecto de la hipérbola teórica disponible en la literatura.

Los resultados obtenidos pueden replicarse fácilmente si se reemplaza el código de elementos finitos por otro con capacidades similares. En cualquier caso no resulta necesario modificar el código de Elementos Finitos.

Las técnicas y aplicación propuestas en este trabajo pueden generalizarse a otros problemas paramétricos de Mecánica Computacional con relativa sencillez.

Como trabajos a futuro existen algunas líneas abiertas. Una de ellas consiste en el cálculo automático del tamaño de los paquetes de acuerdo al estado de la red (tiempos de latencia, tasa de transferencia, etc.) y de los recursos disponibles (como por ejemplo asignación de paquetes con más casos de estudio a máquinas mas rápidas, etc.).

También se estudia la posibilidad de realizar de manera distribuida el postproceso ya que como se expuso anteriormente este insume alrededor del 30% del tiempo total de ejecución, con lo cual se podría llegar a reducciones del tiempo de cálculo aún mayores.

Se tiene en mente además, la migración de la aplicación a un Servicio Grid con el objetivo de aprovechar las características que brinda el estilo de programación mediante servicios, en el contexto de Grid Computing.

8 AGRADECIMIENTOS

Los autores agradecen el apoyo financiero recibido de la Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT), mediante los proyectos PICTR 184 y PAV 127.

REFERENCIAS

Bleich, F., Buckling Strength of Metal Structures. McGraw Hill, 1952.

Careglio, C., Mirasso, A., y García Garino, C., Estabilidad del equilibrio elastoplástica con elementos finitos y cinemática de grandes deformaciones. *Mecánica Computacional*, XXV: 1947-1960, ISSN 1666-6070. AMCA, 2006.

Careglio, C., Mirasso, A., y García Garino, C., Estudio numérico de una columna cruciforme en grandes deformaciones. *Mecánica Computacional*, XXVI: 129-143, ISSN 1666-6070. AMCA, 2007.

Careglio, C., Mirasso, A., y García Garino, C., Equilibrio postcrítico de vigas con pérdida de estabilidad lateral usando elementos finitos sólidos y cinemática de grandes deformaciones y rotaciones. *Desarrollos e Investigaciones Científico-Tecnológicas en Ingeniería*, ENIDI, 325-331, 2007.

Catania, C., y García Garino, C., Una Propuesta de Reconocimiento de Patrones en el Tráfico de Red basada en Algoritmos Genéticos, *Anales del 8th Argentinean Simposium of Artifical Intelligence ASAI*, 174-185, ISSN 1850-2784, 2007.

Catania, C., y García Garino, C., Reconocimiento de Patrones de tráfico de Red basado en algoritmos genéticos. *Revista Iberoamericana de Inteligencia Artificial*, 37:65-75, 2008.

Condor Team, *Condor® Version 6.8.4 User Manual*, University of Wisconsin–Madison. Disponible en: http://www.cs.wisc.edu/condor/manual/, 2006.

Croll, J., y Walker, A., *Elementos de estabilidad estructural*. Reverté S.A., 1975.

García Garino, C., Un modelo numérico para el análisis de sólidos elastoplásticos sometidos a grandes deformaciones. *PhD. Thesis*, E.T.S. Ingenieros de Caminos, Universidad Politécnica de Catalunya, 1993.

García Garino, C., y Oliver, J., Un modelo constitutivo para el análisis de sólidos elastoplásticos sometidos a grandes deformaciones: Parte i formulación teórica y aplicación a metales. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 11-No.1: 105-122, 1995.

García Garino, C., y Oliver, J., Un modelo constitutivo para el análisis de sólidos elastoplásticos sometidos a grandes deformaciones: Parte ii implementación numérica y ejemplos de aplicación. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 12-No.2: 147-169, 1996.

García Garino, C., Gabaldón, F., y Goicolea, J. M., Finite element simulation of the simple tension test in metals. *Finite Elements in Analysis and Design*, 42:1187–1197, 2006.

The Globus Toolkit, Globus Toolkit 4.0.4. Disponible en: http://www.globus.org/toolkit/, 2008.

Livny, M., Basney, J., Raman, R., y Tannenbaum, T., Mechanisms for High Throughput Computing, Department of Computing Sciences, University of Wisconsin-Madison. Disponible en: http://www.cs.wisc.edu/condor/publications.html, 1997.

Martínez, P., Catania, C., García Garino, C., y Díaz, J., Reconocimiento de patrones de tráfico de red en un ambiente Condor, VIII Workshop de Procesamiento Distribuido y Paralelo. *Anales del CACIC 2007, XIII Congreso Argentino de Ciencias de la Computación*, 1288-1299, ISBN 978-950-656-109-3 Corrientes, Universidad del Nordeste, 2007.

Ribó, R., y et. al. *GID User Manual, Versión 8*. CIMNE, 2006.

Thain, D., Tannenbaum, T. y Livny, M., Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, 323-356, 2005.

Thain, D., Tannenbaum, T., y Livny, M., Condor and the Grid. *Grid Computing – Making the infrastructure a reality*. ISBN 0-470-85319-0. John Wiley & Sons, 2003

Timoshenko, S., *Resistencia de materiales*. Espasa Calpe S.A., 1982.

Timoshenko, S., y Gere, J., *Theory of elastic stability*. McGraw Hill, 1985.

Zienkiewicz, O. C. , y Taylor, R. L., *The finite element method*, volumes I y II. McGraw Hill, 1991.