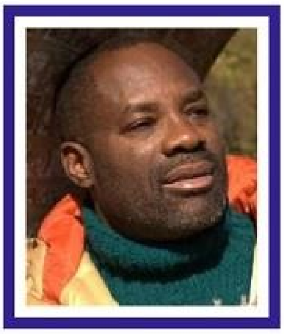
36 Father of the Modern Supercomputer

Adapted from https://www.youtube.com/watch?v=n5qjiS2IISc

https://soundcloud.com/emeagwali/how-i-invented-the-modern-supercomputer-who-is-philip-emeagwali-episode-170120

Adapted from 30:07 minutes
https://www.youtube.com/watch?v=pE2Q8c7bs0w

52:00 minutes
https://www.youtube.com/watch?v=8FjBhSaXKZg&t=4s



Philip Emeagwali
info@emeagwali.com 202-203-8724

Philip Emeagwali Lecture 170928

Visit http://emeagwali.com for complete transcripts of 100+ lectures.
Video: https://YouTube.com/emeagwali
Podcast: https://SoundCloud.com/emeagwali

36.1 How My Supercomputer Discovery Came Together

36.1.1 Who's Philip Emeagwali, the Discoverer of Parallel Processing?

I'm the supercomputer scientist
who was in the news
back in 1989
for experimentally discovering
how and why parallel processing
makes modern computers **faster**
and makes the new supercomputer
the **fastest**,
namely, **the Philip Emeagwali formula**

that then United States President
Bill Clinton
described in his speech of August 26,
2000.
The parallel processing supercomputer
impacts today
and imagines tomorrow.
The fastest, parallel processing
supercomputer
can occupy the space of a **football field**.
But the holy grail in parallel processing
supercomputing
is to compute the fastest
and to do so on the smallest
supercomputer **footprint**.
Parallel processing supercomputers
are used to execute
general circulation models
that are used to
foresee otherwise **unforeseen**
global warming.

The faster the parallel processing
supercomputer
the greater the resolution
of the climate model
or the computational fluid dynamics code
that the supercomputer
is executing.
And the more detailed the model,
the lower the production costs,
and the lower the environmental risks.
And the greater the accuracy
of predicting global warming
that informs and enlightens us
on how to protect
our fragile planet Earth.
And the greater the amount of
crude oil and natural gas that can be
extracted.
The reason one in ten
modern supercomputers were purchased
by the petroleum industry

was that fastest parallel processing
supercomputing
compresses **time-to-solution**
and did so by enabling the
supercomputer modeler
to run more simulations
and making it more efficient
to drill more oil wells
and produce more crude oil and natural gas.
The larger, higher-fidelity
petroleum reservoir simulation
that is executed
on the parallel processing,
high-performance supercomputer
is used to predetermine
the profitability
of newly discovered oil fields
and those of abandoned oilfields,
such as the **Oloibiri** oilfield
of **Bayelsa** state, Nigeria,

1568

that was abandoned in **1978**.
The larger, higher-fidelity
petroleum reservoir simulations
that run on parallel processing
supercomputers
that were purchased
by the petroleum industry
are used to figure out
the most profitable places
to drill for crude oil and natural gas.
My contributions to the use
of the high-performance,
parallel processing supercomputer
to **recover** otherwise **unrecoverable**
crude oil and natural gas
was the cover stories
of top publications
in mathematics, computing,
and petroleum engineering.

36.1.2 My Quest for the Naked Supercomputer

What set me apart
from other supercomputer scientists
was that I was only interested
in naked
parallel processing supercomputers,
or their processors.
Parallel programming ensembles of
up to two-raised-to-power sixteen
processors
and do so for sixteen years
and parallel programming for up to
sixteen hours a day
and parallel programming **alone**
in the United States
was what elevated me
to the fastest and highest levels,
or what some call
the highest computer wizard.
It's the reason American students

are writing school reports
on the contributions of
Philip Emeagwali
to the development of the
modern supercomputer
that computes in parallel
by processing many things (or processes)
at once.
In my experiments
that began as a very vague idea
on June 20, 1974
in the Computer Center,
at 1800 SW Campus Way,
Corvallis, Oregon,
that matured as a discovery
that was reported
on my sixteenth anniversary
of supercomputing,
in the June 20, 1990 issue
of *The Wall Street Journal*.
It was reported by the news media

as a paradigm-shifting discovery
that will change the way we think
about how and why parallel processing
makes modern computers **faster**
and makes the new supercomputer
the **fastest**.
I told the news media
that my contribution to the development
of the new
massively parallel supercomputer
is this:
I **experimentally discovered**
that 65,536 days,
or one hundred and eighty [180] years
of **time-to-solution** on a computer
that is only powered by
only one isolated processor
can be speeded-up to only
one day of **time-to-solution**
across a new internet
that is a global network of

65,536
processors.
That **experimental discovery**
opened the door
to a promising line of research
and my original discovery
have been **experimentally re-confirmed**
as thirty thousand [30,000]
computing-years compressed to
one supercomputing-day.

36.1.3 Core Knowledge Series

My **experimental discovery**
of massively parallel processing
is the core knowledge
of what makes computers
faster
and makes supercomputers
fastest, namely,

the Philip Emeagwali formula
that then United States President
Bill Clinton described
in his White House speech of
August 26, 2000.
To parallel process,
or to compute many things (or processes)
at once
instead of computing
only one thing **at a time**,
is a fundamental knowledge of
modern computer science.
In the 1980s,
I was the lone wolf
supercomputer programmer
of the most massively parallel processing
supercomputer
ever built.
In the 1980s,
I was at the farthest frontier
of massively parallel processing.

In the 1980s,
I was at the crossroad of mathematics,
physics, computing,
and communicating **across**
a new internet
that is a global network of
64 binary thousand
already-available processors
that were married together
by one binary million
regular, short, and equidistant
email wires
and married together
as one cohesive, seamless
supercomputer.
Parallel processing introduces students
to how the modern computer
computes **faster**.
The knowledge of parallel processing
enables students
to participate in conversations about

the development of the computer.
It matters that my contribution
to the development of the fastest
computers
is studied in American schools.
It matters because
eventually, students of today
will be the teachers of tomorrow.
Eventually, teachers of yesterday
will be companions
to the 17th century Isaac Newton.
So, I understood—in the 1970s and ‘80s—
how important it will be
for young black Africans
to see another black African
making a contribution
to the development of the computer.
I discovered that
it was not just for young black Africans
to see me in a leading role
but for old white European scientists

to get accustomed to
a young black African
as their scientific role model.

36.1.4 One Versus Twenty-Five Thousand

The second reason I programmed alone
was because I was the only person
—out of twenty-five thousand [25,000]
supercomputer programmers—
that had the confidence
to communicate and compute what the
textbooks then described
as **impossible**-to-compute.
I was the only person in the world
that understood massively
parallel processing
and understood it
as a small internet
that's a global network of

processors.
I was inspired to conduct research alone
in the decade of the 1980s because
I **invented**
how to harness a global network of
65,536
processors
and optimize that new internet
to yield a speed increase
of a factor of 65,536.

36.1.5 How My Supercomputer Discovery Came Together

For me, my **invention**
of 1989
was the coming together
of mathematical inventions
that began a decade earlier.

Namely, those mathematical inventions
were **encoding** the laws of physics
into the partial differential equations
of calculus
and then **discretizing** those equations
to obtain
a system of equations
of large-scale algebra
and then **coding** that large-scale
algebraic computations
into large-scale floating-point
arithmetical operations
and then **executing** those arithmetical
calculations
within a computer
and, finally, **executing**
those arithmetical calculations across
a small internet
that was my global network of
65,536
commodity-off-the-shelf processors.

The calculus and algebra
is just a way to **encode**
the laws of physics
to enable computational physicists
to **foresee** otherwise **unforeseeable**
global warming,
or **hindcast** or **forecast** the motions
of fluids flowing below,
on, or above
the surface of the Earth
or above the surface
of a distant heavenly body.
In Nigeria and in Africa,
algebra is learned in a **context-less** way.
Even in the United States and in Europe,
only one in a million people
that studied algebra
can explain how algebra
is used to discover and recover
crude oil and natural gas.
For me—**Philip Emeagwali**

—**recovering** previously **unrecoverable** 1580
crude oil and natural gas—and
recovering them
from abandoned oilfields,
such as the Oloibiri Oilfield
of Bayelsa State, Nigeria—
demands that the laws of physics
be **encoded** into a system of dense
partial differential equations
of calculus,
but we encoded them into calculus
without the large-scale algebra;
and demands that
large-scale system of equations
of algebra
be encoded into abstract algorithm,
but we encoded them into
abstract algorithm
without the computation-intensive
floating-point arithmetical operations;
and demands that those

arithmetical operations
be **encoded** into binary **codes**,
but we encoded them into binary **codes**
without the sixteen-bit long
email addresses
that I used, as a lone wolf programmer.
I was the lone wolf
supercomputer programmer
who was in the news headlines
as the “**African Supercomputer Wizard**”
in Los Alamos, New Mexico,
United States.
I forged my technological path
to my new internet
that I visualized
as a new global network of
two-raised-to-power sixteen,
or 64 binary thousand,
processors.
I visualized my new internet
as one cohesive whole unit

that is a new supercomputer
that was outlined by
two-raised-to-power sixteen
already-available
processors
that were married together
by sixteen times
two-raised-to-power sixteen
regular and short email wires
that are equal distances
apart.
The June 20, 1990 issue
of the *Wall Street Journal*
recorded that I—**Philip Emeagwali**—
had invented
how 65,536
processors
could compute together
to solve the toughest problems in
calculus.
I experimentally discovered

the total parallel processing
supercomputing power
of those processors,
not as separate processors,
but as one cohesive unit
that is not a computer *per se*
but that is a new internet *de facto*.
And so on.

36.2 How I Invented the Modern Supercomputer

36.2.1 My Breakout Discovery in Supercomputing

My breakout discovery occurred in 1989.
On the morning that discovery occurred,
I was dressed in blue jeans,
plaid shirt, and white tennis shoes.
I had been sitting silently

and sitting for months in a row
and sitting in front of
my workstation computer
that was tucked away
in a closet-like corner.
I faced three **blank walls**
and I faced an often blank, super-sized
computer monitor.
The **blankness** made it easier for me
to concentrate
on my equations, algorithms, codes,
and emails.
I was a lone wolf programmer
in Los Alamos, New Mexico,
United States,
of a new global network of
64 binary thousand processors.
I was remotely parallel programming
sixteen separate
global network of processors,

each ensemble comprising of
up to two-raised-to-power sixteen
processors.
I visualized each of my ensemble
as my new internet,
or as my new global network of
processors.
It was like silently sitting alone
in a **dark room**
and **interacting** with
65,536 complex machines
that you’ve never seen.
I found it exciting and **cathartic**
inside that **dark sixteen-dimensional**
world.
My breakout discovery
of the massively parallel processing
supercomputer
that occurred on the Fourth of July 1989
in Los Alamos, New Mexico,

United States
was first announced
as a press release in San Francisco,
California.
That press release
that announced my experimental
discovery of parallel processing
was issued and distributed
by The Computer Society
of IEEE.
The Computer Society
is the world’s largest computer society.
Before 1989,
supercomputer textbook authors
explained that parallel processing
supercomputing
—or solving a million problems
(or processes) **at once**,
instead of solving one problem
at a time—

is a beautiful theory
that lacked an **experimental**
confirmation.
Then in 1989,
it made the news headlines
that a lone wolf
African massively parallel processing
supercomputer wizard
in Los Alamos, New Mexico,
United States,
had **experimentally confirmed** that
the **impossible-to-solve**
is, in fact, **possible-to-solve**.
That African supercomputer wizard
experimentally confirmed that
it is possible to solve a million problems
(or processes) **at once**
and solve them while solving
an extreme-scale problem
in computational physics.

I—**Philip Emeagwali**—
is that African supercomputer scientist
that was in the news in 1989.
I **invented**
how to solve the toughest problems
arising in calculus, computing,
and computational physics
and how to solve
such computation-intensive problems
in parallel
and how to solve them across
a new internet.
I **visualized** my new internet
as a new global network of
64 binary thousand,
or two-raised-to-power sixteen,
commonly-available processors.
Or as a global network of
64 binary thousand computers

that are equal distances **apart**
in a sixteen-dimensional universe.

36.2.2 Visualizing Solutions in Hyperspace

Solving the toughest problem
arising in calculus
is akin to playing a complex game
with complex rules
and playing that game
in a sixteen-dimensional universe.
In 1989, it made the news headlines
that I—**Philip Emeagwali**—had
invented
how to play that game
in sixteen-dimensional hyperspace.
I invented
how to execute
floating-point arithmetical operations
and how to do so

at the fastest, parallel processing
supercomputer speeds
ever recorded.
I experimentally discovered that speed
across a new internet
that is my new global network of
two-raised-to-power sixteen,
or 65,536,
commodity-off-the-shelf processors
and that I visualized
as my small copy
of the Internet.
I invented
how to reduce
that computation-intensive
grand challenge problem
to an equivalent set of
64 binary thousand
less computation-intensive problems.
I invented
how to make the **impossible**-to-solve

possible-to-solve.
I invented
how to do the impossible
by synchronizing the email
communications
that I executed across
my global network of
64 binary thousand
processors.
I had to visualize
my **invention-in-progress**
before I invent it.
Often, my visualization
is 90 percent correct.
I discover the remaining ten percent
via experimentation,
or trial-and-error.
I visualized my global network
of processors
as my prototype
for my new internet.

I visualized my large-scale
computational fluid dynamics code
as a computation-intensive game
that I had to play
in a sixteen-dimensional hyperspace
and play **across**
a fifteen-dimensional chess board.
I visualized my large-scale
general circulation model
as a computational physics code
that was comprised of
64 binary thousand atmospheres
that had a **one-to-one**, nearest-neighbor
correspondence
with my as many processors.
I visualized my processors
as 65,536 fifteen-dimensional squares
on that sixteen-dimensional chessboard.
I visualized
the hyper-spatial arrangements
of my sixteen-dimensional pieces

and I visualized
the configurations of my chessboard
in hyperspace
and I visualized those configurations
as changing after every move.
I visualized solving the toughest problem
in calculus—namely, the largest-scaled
computational physics codes—
as comprising of
64 binary thousand blocks of
atmosphere.
I visualized each block
as containing flowing air and moisture,
or fluids that are in deterministic
motions
in sixteen-dimensional **hyperspace**-time.
I visualized
how to parallel process in hyperspace.
Parallel processing is the *sine qua non*
of modern computing.
Parallel processing is the essential

1594

condition
for the modern computer
and the technology
that is absolutely necessary
for the modern supercomputer.
Parallel processing is the crucial,
indispensable, and disruptive technology
for extreme-scale computational
physicists
and mathematicians.
Without parallel processing,
the world’s fastest supercomputer
will take 30,000 years
to compute what it now computes
in only one day.
The invention of parallel processing
fueled the growth of
computational science.
As the sole and only full-time
programmer
of that ensemble of

two-raised-to-power sixteen processors,
I gave myself the permission
to break every rule
in that sixteen-dimensional hyperspace,
except that my fluids in motion
could not violate the laws of physics
that I encoded
into my system of coupled, non-linear,
time-dependent, and state-of-the-art
partial differential equations
of calculus.

36.2.3 My Eureka Moment

My Eureka Moment
was in inventing
those global network of
65,536 tightly-coupled processors
with each processor
operating its own operating system
and with each processor

having its own dedicated memory
that shared nothing with each other
and in inventing that ensemble
as a new internet
and in inventing those processors
as having a one-to-one correspondence
to the as many vertices of the cube
in the sixteenth dimensional universe
and in inventing that cube
as **tightly circumscribed**
by a sphere
that is also in the sixteenth
dimensional universe
and in inventing a global network of
processors
that define and outline a new internet.
That invention
was the beginning of my realization
that the computer and the internet
could become like identical twins.
That invention was **visceral**.

After my **experimental discovery**
and after the news headlines
that followed it
I became like the ancient mariner
who travelled around the world
to tell his tales to different people.
Like the ancient mariner,
I'm here to tell you my tales.
I'm here to share lessons
that I learned as a lone wolf
at the farthest frontier of technology.
I'm here to help you
cross new frontiers
and discuss how we can conquer
today's grand challenges.

36.3 My Contributions to Computational Mathematics

36.3.1 A British-Protected Child

I began my journey
to the unknown world of supercomputers
in Akure,
in the heart of Yoruba Land
in the then British West African colony
of Nigeria.
I began my journey on August 23, 1954,
my birthdate.
In the 1950s,
the flag of Nigeria
was the Union Jack.
In the 1950s,
the Governor-General of Nigeria
was a non-Nigerian, a British,
that was appointed by the Queen of
England.
In the 1950s,
the currency of Nigeria
was the British West African pound.
In the 1950s,

1599

my British West African travel passport
would have described me
as a British-Protected Child.
I began my journey along a small road
in Akure
that was named **Okemeso** Street
and the sixth person
in a tiny Boy’s Quarter
that was at the intersection
of **Okemeso** Street
and **Oba Adesida** Road,
Akure.
I began my journey with a dim lamp
and at a time
the word “computer”
was not in the vocabulary
of any Nigerian newspaper.

36.3.2 A Computational Mathematician in Oregon

1600

I programmed sequential processing supercomputers
on June 20, 1974 in Corvallis, Oregon.
Back in 1974, I programmed
sequential processing supercomputers
as a hobby, not for a career.
Ten years before I programmed
sequential processing supercomputers,
no university in the United States
had a computer science department.
The field of computer science, itself,
was a late 1940s outgrowth
from the computation-intensiveness
of the numerical solution
of the ordinary differential equation
of modern calculus.
Most ordinary differential equations

encoded
the Second Law of Motion
of physics.
That Second Law of Motion
governs the motions of a missile
or a spacecraft.
Since 1946,
the supercomputer silently consumed
the most large-scale system of equations
of algebra.
A modern example
of the most large-scaled system of
equations
are those that arose from discretizing
partial differential equations.
The reason computing
reasonably accurate solutions
of a partial differential equation
is computation-intensive

is that computing it
required an infinite number of
calculations.
Therefore, it will take forever
to compute the exact answers
for an initial-boundary value problem
for which a system of
partial differential equations
holds in its interior
and its specified initial
and boundary conditions
hold on the exterior.
In formal mathematical lingo,
the exact solution
of the initial-boundary value problem
is defined across infinite points
in space and time.
Solving the companion
and approximating system of
partial **difference** equations

of algebra
is computation-intensive because
the partial **differential** equations
within the system
are coupled, non-linear,
time-dependent, and hyperbolic.
In a literal sense,
an initial-boundary value problem
such as the problem
at the mathematical core
of the general circulation model
that is used to **foresee** otherwise
unforeseeable
global warming
is a boundary value problem.
It's **inner boundary**
is the nearly eight thousand mile
diameter
surface of the Earth.
It's **outer boundary**

1604

is the uppermost atmosphere
that encircles our planet Earth.
Constructing the general circulation
model that will be used
to **foresee** otherwise **unforeseeable**
global climate change
and executing that model across
a new internet
that is a global network of processors
is far more complicated
than solving a textbook
partial differential equation.
The initial-boundary value problem
that is at the calculus core
of petroleum reservoir simulation
is far more complicated
than the **well posed elliptic**
partial differential equations
that I learned in the mid-1970s
and learned from mathematicians

in Kidder Hall
at 2000 SW Campus Way
in Corvallis, (Oregon), United States.
Kidder Hall
was only 200 feet away
from two of the world's fastest
sequential processing supercomputers
that I was then programming.

36.3.3 Contributions to Computational Mathematics

I am giving this lecture today
because of my experimental discovery
of parallel processing
that made the news headlines
and did so within
the mathematics community
back in 1989.
I **invented**
how to look beyond the storyboard,

look beyond the blackboard,
look beyond the motherboard,
and look beyond all three boards
to **experimentally solve**
partial **difference** equations
of algebra
that arose from discretizing
and approximating
partial **differential** equations
of calculus
that encoded a law of physics,
such as the Second Law of Motion.
I **invented**
how to solve such initial-boundary value problems
and how to solve them **across**
a new internet that I invented
and constructively reduced to practice
as my new global network of
65,536 tightly-coupled processors
that shared nothing with each other.

In that global network,
my processors were identical
and were equal distances **apart**.

36.3.4 Foundations for Computational Mathematics

I emigrated from Onitsha (Nigeria)
to Oregon (United States).
I last lived in Africa
when I was nineteen years old.
I first came to the United States
on Sunday March 24, 1974.
I spent my first night
in the United States,
alone, and in Room 36
of Butler Hall,
Monmouth, Oregon.
On my desk in Butler Hall
was a 568-page blue hardbound book
that was titled:
[quote]

“An Introduction
to the Infinitesimal Calculus.”
[unquote]
That calculus book
was written by G.W. [George William] Caunt
and published by Oxford University Press.
I acquired that calculus book
through an act of serendipity.
In June 1970 and five months
after the Nigeria Biafra War ended
and in Christ the King College,
Onitsha (Nigeria),
I was given the nickname “Calculus.”
They called me “calculus”
because it seemed like
I carried that calculus book
at all times.
In 1971 and ‘72,
I studied calculus independently
and in the late afternoons
at Sacred Heart Primary School,

1609

Ibuzor, Mid-West State, Nigeria.
In the early 1970s,
I imagined that calculus book
as my magical window
that enabled me to study calculus
by correspondence
and through a text-only version
of a calculus lecture
given by the author Professor G.W. Caunt
and given at the University of Oxford,
England.
As a 15-year-old in June 1970,
I gained a glimpse
of the frontier of calculus.
That awareness
inspired me to begin my quest
for new calculus
that could only be discovered
in the *terra incognita* of calculus.
I took two decades,
onward of June 1970,

1610

to arrive at the frontiers
of calculus and algebra—namely,
invent **Philip Emeagwali**’s
partial differential equations
that are defined
in the interior of the domain
of an initial-boundary value problem
and discretize those equations
into millions upon millions
of approximating
partial difference equations
of algebra
and, finally, to experimentally discover
how to solve such computation-intensive
algebraic problems
and solve them
at the fastest, parallel processing
supercomputer speeds.
Such boundary value problems
are at the mathematical core
of general circulation models,

at the mathematical core
of petroleum reservoir simulators,
and at the mathematical core
of large-scale
computational fluid dynamics.
Such boundary value problems
gave rise to the algebraic approximations
of the partial differential equations
that I translated for the processor.
As a research computational mathematician,
I discovered that
the new frontier of calculus
is across a new internet
that is a global network of
64 binary thousand processors.

36.4 My Contributions to Calculus

36.4.1 Parallel Computing the Toughest Problems

For the twenty years, onward of June 1970,
I studied calculus,
beginning with the book by G.W. Caunt,
and did so almost daily
and ending the new calculus
that comprised of 36 partial derivative terms
that I invented
in the early 1980s.
In the early 1970s and in **Onitsha**, Nigeria,
I studied calculus on the storyboard
in which the laws of physics
were the stories
and studied calculus
as the most powerful instrument of physics.
In the mid-1970s
and in Kidder Hall, **Corvallis**, Oregon,
United States,
I continued to study calculus
and studied it as differential equations
on blackboards that were
200 feet away from

two of the fastest supercomputers
in the world.
In the late 1970s
and in the Foggy Bottom neighborhood
of **Washington**, DC,
I continued to study calculus
and studied it as advanced expressions
called partial differential equations
that encoded the laws of physics
and I studied the calculus
that was in the **granite core** of
extreme-scale computational fluid dynamics.
Throughout the 1980s,
I continued my quest for new calculus
at the frontier of calculus
and I invented
how to solve the toughest problems
in calculus
and how to solve them **across**
64 binary thousand
identical processors.

As a large-scale
computational mathematician
of the 1980s,
my contributions to calculus
were two-fold.
My first contribution to calculus
that made the news headlines
in 1989
was my invention
of how to solve a million problems
(or processes) **at once**
and how to solve them
as solving initial-boundary value problems
of calculus
and solving them across
64 binary thousand processors
that are equal distances **apart**
and that are identical.
The reason my invention
made the news headlines in 1989
was that it opened the door

to the modern, parallel processing
supercomputer
that computes across
ten million
six hundred and forty-nine thousand
six hundred [10,649,600]
identical processors.
My second contribution to calculus
that also made the news headlines
in 1989
was my invention
of how to solve the
initial-boundary value problems
of calculus and of crude oil and natural gas
recovery
and how to solve them better and faster.
I **invented** 36 partial derivative terms.
My derivative terms
expanded the existing
45 partial derivative terms.
My total of 81

partial derivative terms
are the key components
of nine partial differential equations
that are also known as the
Philip Emeagwali’s equations.
That system of nine
coupled, non-linear, time-dependent, and
state-of-the-art **Philip Emeagwali**’s
equations
governs the three phase flows
of crude oil, injected water, and natural gas
in the x-, y-, and z-directions.
Those 81 derivative terms of calculus
define more accurate,
larger, higher-fidelity
petroleum reservoir simulation models.
Using 81, instead of 45,
partial derivative terms
enables the petroleum geologist
do her calculus better and faster.
My 36 partial derivative terms

were written in the *lingua franca*
that is unfamiliar
to those lacking expertise
in calculus.
My 36 partial derivative terms
encoded **familiar** inertial forces.
I made the **familiar**
unfamiliar.
Those 36 partial derivative terms
were for simulating the flows
of crude oil, injected water, and natural gas
across oilfields.
My mathematical terms
encoded the temporal
and the convective inertial forces
that **drive** the three-dimensional
motions of crude oil, injected water,
and natural gas
and **drive** them across an oilfield
and **drive** them
from water injection wells

1618

to production wells.
As an extreme-scale computational physicist
of the 1980s, calculus
was always at the core
of my computer codes.
As the massively parallel
supercomputer scientist
that was at the farthest frontier
of the fastest computing
of the 1980s, calculus
was always at the core
of my 64 binary thousand computer codes.
I emailed those computer codes
with **one-to-one** correspondence
to 64 binary thousand
identical processors.
Those plentiful, powerful, and inexpensive
already-available processors were married
together
by a new global network of
one binary million regular and short

email wires that were equal distances
apart.
Calculus began 330 years ago
and many research mathematicians
contributed to the development of calculus.
My contributions to calculus
were front page stories
of top mathematics publications,
such as the cover story
of the May 1990 issue
of the *SIAM News*.
The *SIAM News* is written by
mathematicians for mathematicians.
My two contributions to calculus
are these:
First, I expanded the calculus
of crude oil and natural gas recovery
by 36 partial derivative terms
that enhanced the accuracy
of a system of nine coupled, non-linear,
time-dependent, and state-of-the-art

partial differential equations
that I invented
for high-fidelity
petroleum reservoir simulations
within a multi-disciplinary environment.
Second, I invented
how to solve the system of
millions upon millions of approximating
partial **difference** equations
of algebra
that arose from my system of
partial **differential** equations
of calculus
and how to solve them **across**
an ensemble of 64 binary thousand
processors
that are married together
as a new internet
and married
by one binary million email wires.
The terminology

“partial differential equation”
[par·tial dif·fer·en·tial e·qua·tion]
was coined in **1845** to describe an equation
that contained partial derivatives.
Seventeen decades later,
the partial differential equation
is widely used by physicists, chemists,
biologists, economists, and engineers.
In physics, the partial differential equation
is used to model the motions of fluids
that enshroud the Earth,
such as atmospheric circulation models
above the surface of the Earth,
ocean circulation models
on the surface of the Earth,
and petroleum reservoir models
below the surface of the Earth.
The Earth doesn’t fit into a lab,
or into one computer.
For that reason, I invented
how to fit

the Earth into a new supercomputer
that is not a computer *per se*
but that is a new internet *de facto*.
My new internet
is powered by an ensemble of
65,536 commonly-available processors
with each processor
operating its own operating system
and with each processor
having its own dedicated memory
that shared nothing with each other.

36.4.2 Changing the Way We Calculate in Calculus

The calculus I learned
in June 1970 and at age 15
differs greatly from the calculus
that I invented twenty years later.
After a decade of research
at the frontiers and crossroad
of calculus and computing,
I invented how to use

parallel processing supercomputers
to solve the toughest
problems in calculus.

To invent a new supercomputer
that solves the toughest problems
in calculus
is to make the **impossible**-to-solve
possible-to-solve.

My **mathematical discovery**
was newsworthy
and made me the cover story
of top mathematics publications.
I was the cover story
of the June 1990 issue
of the *SIAM News*
that is the flagship publication
of the Society for Industrial
and Applied Mathematics.
The reason my invention
was cover stories

of mathematics publications 1624
was that I invented
how to harness
parallel processing technology
to solve the toughest problems
in calculus.
In the old way,
we unsuccessfully tried to solve
the toughest problems
in calculus
on the blackboard or motherboard
and failed to solve them
on one processor.
In the new way,
we can solve
the toughest problems
in calculus
and solve them across
up to ten million
six hundred and forty-nine thousand
six hundred [10,649,600]
identical processors.

36.4.3 I Discovered the Impossible is Possible

During those twenty years
—onward of June 1970—
my mathematical and scientific maturity
grew as expected
of a mathematical scientist
that devoted twenty years
to his craft
and **searching** for new calculus
at the frontier of abstract calculus
and **searching** for new algebra
at the frontier of large-scale algebra
and **searching** for the fastest
floating-point arithmetical operations
at the frontier
of the most massively parallel
supercomputer
ever built

and that is a global network of
64 binary thousand processors
and that is a new internet.
In the seventh year
of that twenty-year sojourn
to the farthest frontier of computing,
I drifted and became an astronomer
who was primarily interested
in distant galaxies in outer spaces.
But in later years,
I drifted from outer space
in the third dimension
that contained invisible **black holes**
to inner mathematical spaces
in the sixteenth dimension
where finding the supercomputer
is like searching for a black goat
at night.
The reason I discovered that
the **impossible**-to-compute
is **possible**-to-compute

is that the toughest problem
that is impossible
to solve in ten years
could be possible
to solve
in twenty years.
The research mathematicians
that preceded me were attempting to use
their **ten years** of training
to solve the toughest problem in calculus
that I solved after **twenty years**
of training.
I trained for **twenty years**
in the mathematical
and computational sciences
before I became cover stories
for mathematicians.
The first programmable supercomputer
was produced in 1946
at **Aberdeen Proving Ground**,
Aberdeen, Maryland, United States.

The toughest problem
for that first supercomputer
was to numerical solve
the **ordinary differential equations**
of calculus
that was used at **Aberdeen Proving Ground**
of Maryland
and used to compute the trajectories
of missiles.
In the mid-1970s and at Kidder Hall,
Corvallis, Oregon, United States,
I was simultaneously studying
how to solve an
ordinary differential equation
on the blackboard
and sequentially programming
two of the fastest supercomputers
in the world.
Those two supercomputers
were 200 feet away
from Kidder Hall.

I used the **Teletypewriter Model 33 ASR**
that was inside Kidder Hall
to log into each supercomputer.
In Kidder Hall, I discovered that
the numerical solution
of the **ordinary differential equation**
of calculus
is not as computation-intensive
as the numerical solution
of the partial differential equation
of calculus.
Before my arrival on Sunday March 24, 1974
in Monmouth, Oregon, United States
and back in the late 1940s, ‘50s, and ‘60s
the **ordinary differential equation**
of calculus
was solved within the supercomputer
that helped send men to the moon.

36.5 How I Became a Polymath

1630

Sixteen months
after the last man returned from the moon,
I programmed supercomputers
in Corvallis, Oregon, United States.
I programmed supercomputers
on June 20, 1974 and at age nineteen.
Three weeks
after **I programmed** supercomputers
I was on the cover
of a local newspaper
that circulated in the cities of
Monmouth
and Independence, Oregon, United States.
And **I programmed** supercomputers
at a time
mathematical physics
was being replaced
by far more powerful computational physics,
such as **general circulation models**
that are used to foresee
otherwise unforeseeable global warming.

Computational physics
is where physics entered into engineering
to become useful.
I programmed supercomputers
when extreme-scale computational physics
was paradigm shifting
from sequential processing supercomputers
that operated only on **pairs of numbers**
and paradigm shifted
to vector processing supercomputers
that operated on **pairs of lists**
of numbers.
I programmed
sequential processing supercomputers
in 1974
and **I programmed** them
because in 1974
parallel processing was ridiculed
as a beautiful theory
that lacked experimental confirmation.
In the 1980s,

there were twenty-five thousand
supercomputer scientists
that were programming
vector processing supercomputers.
But I—**Philip Emeagwali**—was the lone wolf
that programmed
the most massively parallel
processing supercomputer
ever built.
My breakout work
was the discovery of how to harness
the slowest 65,536
processors
that each performed
47,303 calculations per second
and performed at that slow speed
to attain the world's fastest speed
in computation
of 3.1 billion calculations per second
that made the news headlines
in 1989.

Eleven years later,
and in a White House speech televised
on August 26, 2000,
my invention
of how to push the speed limits
of the supercomputer
was, again, extolled
by then President Bill Clinton.
That massively parallel processing
supercomputer
that I **experimentally discovered**
as a new global network
is the pre-cursor
to the modern supercomputer.
As an inventor, my contribution
to the development
of the fastest supercomputers
was to invent **something** from **nothing**.
I invented the modern supercomputer
from yesterday's computer.
I invented a new internet

that is a global network of
64 binary thousand computers.
I invented that new internet
from singular computers.
I **experimentally discovered**
that parallel processing,
or solving 64 binary thousand problems
at once
instead of solving one problem
at a time,
is not a huge waste of everybody’s time.
The inventor **experimentally discovered**
that the impossible is, in fact, possible.

36.5.1 Calculating in Parallel

The reason parallel processing
was **dismissed**
as a huge waste of everybody’s time
was that the supercomputer

that did many things (or processes)
at once
was **counter-intuitive**.
The computer
was invented by humans
and in the image of humans.
The mathematician
visualized only one human computer
solving his initial-boundary value problems—
such as the general circulation models
that are used
to foresee otherwise unforeseeable
global warming.
The mathematician
visualized only one human computer
solving her
initial-boundary value problems
and solving them alone,
or in sequence
and not solving them in parallel.
The polymath

thinks beyond the laws of physics
on his storyboard,
thinks beyond the calculus,
on his blackboard,
thinks beyond the algebra
on his motherboard,
and thinks beyond the computer codes
that he must email
across his 64 binary thousand
motherboards.
The polymath
thinks around a globe
in the sixteenth dimension.
The polymath
visualizes his internet
as encircling a globe that is **a small copy**
of the Earth.
The polymath
visualizes his internet
as a global network of
two-raised-to-power sixteen,

or 65,536,
identical processors.
The polymath
visualizes his internet that encircles
a room-sized globe
as **a small copy** of the internet
that encircles
the planetary-sized Earth.
In the 1970s, I visualized
those 64 binary thousand processors
as the 64 binary thousand electronic brains
of my **HyperBall** supercomputer.
And I visualized those electronic brains
as **equidistantly** distributed
around the **fifteen-dimensional** surface
of a globe, or a hyperball,
and distributed
in a **sixteen-dimensional** universe.
As a trained geometer,
it was easy for me to visualize this internet.
I visualized the **uniformity** and **regularity**

that was needed to understand
how to parallel program
my 64 binary thousand processors.
I visualized
how to parallel program those
plentiful, powerful, and inexpensive
already-available processors
via self-relative email communications
and parallel program them
to and from sixteen
mutually orthogonal directions.
But harnessing the power
of two-raised-to-power sixteen processors
was not easily imagined
by every day engineers
who were trained
to think in only three dimensions.
My invention of parallel processing
was rejected, in part, because
I was doing everything
they were trained not to do, namely, to do

1639

65,536 things
at once,
instead of doing only one thing
at a time.
I invented
how to solve the toughest problems
in computational mathematics
and computational physics
—problems such as
using larger, higher-fidelity
petroleum reservoir simulator
to **discover** and **recover**
otherwise elusive
crude oil and natural gas—and I invented
how to make the impossible-to-compute
possible-to-compute
and I invented
how to compute them **sixteen** dimensionally
and along **sixteen**
mutually orthogonal directions.
I invent**ed**

1640

how to solve
the **initial-boundary value problems**
of calculus
and solve them across
64 binary thousand
processors.
The **initial-boundary value problems**
that I experimentally solved
encoded a set of laws of physics
and encoded them
into a system of
partial differential equations
of calculus.
Each partial differential equation
governed a physical phenomenon.
A physical phenomenon
might be the large-scale motion
of air and moisture
within the atmosphere of the Earth.
In atmospheric modeling,
such as weather forecasting

or **general circulation modeling**,
the interior of my
initial-boundary value problem
will correspond to the Earth’s atmosphere.
In the late 1940s, ‘50s, and ‘60s,
initial-boundary value problems
of calculus
were approximated and reduced to
large-scale systems of
partial **difference** equations
of algebra.
Those systems of equations
arose from
the finite difference
and/or the finite element discretizations
of the governing
partial **differential** equations
of the initial-boundary value problem.
Those systems of equations
were solved on supercomputers
that were powered by

only one isolated sequential processing unit
that was not a member
of an ensemble of processors
that communicates and computes
together
and as one seamless, cohesive
supercomputer.
In the 1970s and ‘80s,
initial-boundary value problems
were solved on supercomputers
that were powered by
only one isolated vector processing unit
that was not a member
of an ensemble of processors.

36.5.2 Wanted: A Polymath

There’s no school of genius students
learning from genius teachers.
Our genius resides within us.

As a lone wolf supercomputer scientist,
I had to be a polymath
to be able solve the toughest problem
of supercomputing
and solve the problem alone.
I had to be a polymath
to understand
the set of laws of physics
and understand those laws
as my lowest common denominator.
I had to be a polymath
to translate the toughest problem
in computational physics
and translate it alone
and translate it
from the frontier of knowledge
of extreme-scale computational physics
to the frontier of knowledge
of the partial differential equations
of calculus,
to the frontier of knowledge

of extreme-scale algebra, 1644
and to the frontier of knowledge
of massively parallel supercomputing.
I had to be a polymath
to translate
a grand challenge problem alone
and translate it
across uncharted territories
of technological knowledge
where I recorded unrecorded speeds
in computation.
That uncharted territory
comprised of
a global network of
the slowest 65,536
processors
that were equal distances
apart
that computed together
to emulate the fastest supercomputer.
I had to be a polymath

to translate the grand challenge problem
alone
and translate it
from physics to algebra to calculus
and translate it
back to algebra and to arithmetic
and translate it
into a processor
and translate it
through a new internet.
I had to be a polymath
to invent that new internet alone
and invent it
as a global network of
64 binary thousand
processors.
I had to be a polymath
to deeply understand
and to clearly visualize
in the sixteenth dimension
how my seamless emailing

of two-raised-to-power sixteen,
or 64 binary thousand, emails
will save me from
the 64 binary thousand **square corners**,
with a one-to-one correspondence
with my as many processors.
I had to be a polymath
to deeply understand
how the sixteen times
the two-raised-to-power sixteen,
or the one binary million,
unique arrangement of zeros and ones
will save me from
the one million forty-eight thousand
five hundred and seventy-six [1,048,576]
bi-directional **sharp edges**
with a one-to-one correspondence
with my as many email wires.

A fifth grader doing a school report
on Philip Emeagwali asked:

"Are you a black genius?"
I answered:
"Is Albert Einstein a Jewish genius?"
Genius is not a white trait.
Nor is it a black trait.

Genius is a human trait!
The genius
is the ordinary person
that found the extraordinary
in the ordinary.

36.5.3 How I Became a Polymath

Back in the 1980s,
they were 25,000 supercomputer
programmers
in the United States alone.
Each supercomputer scientist programmed
a vector processing supercomputer.
I—**Philip Emeagwali**—was the lone wolf

fulltime
supercomputer scientist
that was at the farthest frontier
of the most massively parallel
supercomputer.
That parallel processing machine
was the **pre-cursor**
of the modern supercomputer
of today
that computes in parallel
and communicates across millions
of processors.
To some extent,
that parallel processing machine
was the **pre-cursor**
of the modern computer
of today
that computes in parallel
and communicates synchronously
and do both across hundreds of
processors.

I didn’t **become a polymath**
with the help of an instructional DVD.
I didn’t arrive overnight
at the farthest frontiers
of human knowledge.
And I didn’t **become a polymath**
that arrived at the uncharted territory
of supercomputing
and arrived there
by enrolling in a six-day coding school.
I became a polymath
after two decades of training
that was onwards of June 1970,
and the date I began studying calculus.
I began programming supercomputers
at 1800 SW Campus Way,
Corvallis, Oregon, United States
on June 20, 1974
at age nineteen.
I became a polymath
that is a supercomputer scientist

after a decade and half
of parallel programming the fastest
supercomputers.
In the 1980s,
I was the lone wolf programmer
of the most massively
parallel supercomputer
ever built.
I became a polymath
and a computer wizard
after and because
I had programmed
more processors
than any person
that ever lived.

36.5.4 The Lone Wolf at the Farthest Frontier

I programmed a supercomputer
nearly every day

and programmed sixteen supercomputers
in sixteen years
before I became a supercomputer scientist.
You need discipline
more than you need talent
to become a supercomputer wizard.
In my fifth decade of supercomputing,
that is onward of June 20, 1974,
I accumulated a body of inventions
to draw from.
I had to re-examine my body of discoveries.
After five decades,
the context of my discovery
is **different**.
And I am also **different**.
I am selecting from facts
and truths
that I hope will remain **timeless**
and **evergreen**.
Unlike four and half decades ago,
I now possess a third eye

that sees into the sixteenth dimension.
In hindsight, I realized that the
toughest problem
in massively parallel processing
that I solved chose me
rather than me chose the problem.
I have more materials
to **contextualize** my supercomputer
experiments
of the preceding four and half decades.
In the 1970s and ‘80s,
I was punished and ostracized
for challenging the central dogma
of the supercomputer world
that demanded
only one **isolated** processor
that was not a member
of an ensemble of processors.
Those research supercomputer scientists
that were risk averse
and that were merely seeking a factor of

two percent increase
in supercomputing speed
were handsomely rewarded
while I was punished
for seeking a factor of
65,536 fold increase
in supercomputer speed.
I was called a lunatic
when I advocated
massively parallel processing.
In November 1982,
I gave a lecture
on massively parallel processing.
I gave the lecture in a conference auditorium
that was a short walk
from The White House
in Washington, DC.
I gave my lecture
on how to massively parallel process
65,536 **initial-boundary value problems**
and on how to process them **at once**,

or how to solve the toughest problems
in calculus
and solve them
across as many plentiful, powerful,
and inexpensive commodity
off-the-shelf processors,
instead of solving them **in sequence**
and solving them within only one
isolated processor
that was not a member
of an ensemble of processors
that communicates and computes
together
and as one seamless, cohesive
supercomputer.
Because parallel processing
was then—in the 1970s and ‘80s—regarded
as a huge waste of everybody’s time,
only one young
computational mathematician
attended my **November 1982** lecture

on how to massively parallel process
the toughest problems in mathematics.
In the 1970s,
the *Computer World*
was the flagship publication
of the computer industry.
And the **National Computer Conference**
was the largest computer conference
in the world.
The June 14, 1976 issue
of the *Computer World*
interviewed
the foremost supercomputer experts
that attended the 1976
National Computer Conference.
Based on that interview
the *Computer World*
wrote a state-of-the-art article titled:
[quote]
“Research in Parallel Processing
Questioned as ‘Waste of Time.’”

[unquote]
In the 1980s, only one person
was at the farthest frontier
of massively parallel supercomputing.
I was the only fulltime programmer
of the most massively parallel
supercomputer
of the 1980s.
In the 1970s and ‘80s,
the leaders of thought
in vector processing supercomputing
ridiculed parallel processing
and dismissed it
as a beautiful theory
that lacked experimental confirmation.
That pessimism
towards parallel processing
was the reason
I—**Philip Emeagwali**—was the only
fulltime programmer
of the most massively parallel processing

supercomputer
of the 1980s and earlier.
I was alone at the farthest frontier
of the most massively parallel
supercomputer
ever built.
That fastest supercomputer
of the 1980s
is the **pre-cursor**
to the modern supercomputer
that is the fastest computer
of today.
My 1989 experimental discovery
of parallel processing
was not just about supercomputing
64 binary thousand times **faster**.
That discovery
made the news headlines because
it was about making **possible**
65,536 solutions
that were otherwise **impossible**.

1658

In 1989, to invent a supercomputer
was to make the **impossible**-to-compute
that was impossible
with vector processing
supercomputer technology
possible-to-compute
with parallel processing
supercomputer technology.
In the future, to invent a supercomputer
will be to make the **impossible**-to-compute
that is impossible
with parallel processing
supercomputer technology
possible-to-compute
with an as-yet-to-be-invented technology.