Αρχιτεκτονική ενίσχυση αγροτικών IoT συστημάτων πραγματικού χρόνου

Δημήτριος Ντέντας

19 Μαΐου 2025

Περιεχόμενα

1
Εισαγωγή

2

Επισκόπηση της Ερευνητικής Περιοχής

Στη σύγχρονη γεωργία, ένας δυσλειτουργικός αισθητήρας που περνά απαρατήρητος για ώρες μπορεί να οδηγήσει σε αποτυχημένο πότισμα ή καταστροφή καλλιεργειών. Αυτό το ενιαίο σημείο αποτυχίας αναδεικνύει την ανάγκη για αυτόνομα, ανθεκτικά και παρατηρήσιμα συστήματα σε κλίμακα. Η ανάγκη αυτή εναρμονίζεται με τον οδικό χάρτη (Roadmap) προς ανθεκτικά Κυβερνοφυσικά Συστήματα (CPS) που περιγράφεται από τους Ratasich et al. [1], οι οποίοι δίνουν έμφαση στην ανίχνευση ανωμαλιών κατά τη διάρκεια λειτουργίας, στην απομόνωση σφαλμάτων και στην αυτοΐαση σε δυναμικά Internet of Things (IoT) περιβάλλοντα.

Οι συσκευές IoT παράγουν, σε πραγματικό χρόνο, μεγάλο όγκο δεδομένων, είτε σε μορφή χρονοσειράς, είτε ως ιστορικά δεδομένα [2]. Επομένως, η κλίμακα αυτή αποτελεί βασική αιτία εμφάνισης προβλημάτων και αστοχιών. Ωστόσο, για να εξασφαλιστεί η ομαλή λειτουργία των κρίσιμων υποδομών ενός νευραλγικού τομέα, όπως η γεωργία, οφείλουμε ως μηχανικοί να παρέχουμε λύσεις για τον παραγωγό και κατά συνέπεια για τον πολίτη, που εξασφαλίζουν τόσο την ποιότητα, όσο και την αναμενόμενη ποσότητα των αγροτικών προιόντων.

Σε χώρες όπως η Ελλάδα, όπου η οικονομία και η αυτονομία έγκεινται σε μεγάλο βαθμό στη γεωργική παραγωγή, η υστέρηση του γεωργικού τομέα ως προς την ενσωμάτωση τεχνολογικών καινοτομιών έχει

συμβάλει στην αποδυνάμωσή του και στη χαμηλή του συμβολή στην οικονομική ανάπτυξη, όπως επισημαίνουν οι Kyrkilis et al. [3]. Η αξιοποίηση τεχνολογιών πραγματικού χρόνου θα μπορούσε να αποτελέσει κρίσιμο παράγοντα για την αντιστροφή αυτής της τάσης και τη διασφάλιση της βιωσιμότητας και της αποδοτικότητας του πρωτογενούς τομέα.

Σε κρίσιμες εφαρμογές, όπως η αγροδιατροφή, η διατάραξη στη ροή δεδομένων ή στη λήψη των αποφάσεων μπορεί να οδηγήσει σε σπατάλη τροφίμων, οικονομική απώλεια ή επισιτιστική ανασφάλεια. Σύμφωνα με τους Callo και Mansouri [4], η ανθεκτικότητα των παγκόσμιων δικτύων διανομής τροφίμων εξαρτάται από την δυνατότητα των πληροφοριακών συστημάτων να ανταπεξέρχονται σε γεωπολιτικές ή υγειονομικές κρίσεις μέσω μηχανισμών ευελιξίας και προσαρμογής. Αυτό σημαίνει πως σε ότι αφορά σε αντίστοιχα συστήματα, στα οποία βασίζεται ο πληθυσμός (και η οικονομία) μιας χώρας, η αρχιτεκτονική που χρησιμοποιείται θα πρέπει να είναι άρτια, μελετημένη, αλλά και να εξασφαλίζει την ομαλή και συνεχή λειτουργία τους.

Η σύγχρονη ερευνητική κοινότητα έχει μετατοπίσει το ενδιαφέρον της από στατικές προσεγγίσεις παρακολούθησης προς ολοκληρωμένες αρχιτεκτονικές λειτουργικής ανθεκτικότητας, οι οποίες συνδυάζουν συνεχές monitoring, αυτόματη διάγνωση βλαβών και δυναμική ανάκαμψη. Βασισμένα στις αρχές των CPS, τα σύγχρονα αυτά πρότυπα επικεντρώνονται όχι μόνο στην αποτροπή σφαλμάτων, αλλά και στην ικανότητα του συστήματος να απορροφά, να ανακάμπτει και να προσαρμόζεται σε απρόβλεπτες συνθήκες λειτουργίας, με διατήρηση της κρίσιμης απόδοσης. Όπως επισημαίνουν οι Lee et al. [5], η ποσοτικοποίηση της ανθεκτικότητας απαιτεί νέες μετρικές, όπως η καμπύλη ανθεκτικότητας (resilience curve) και το πλαίσιο R4 (Redundancy, Robustness, Resourcefulness, Rapidity), τα οποία μπορούν να ενσωματωθούν ακόμα και σε πραγματικά περιβάλλοντα cloud-native, διασφαλίζοντας ιδιότητες όπως υψηλή διαθεσιμότητα, αυτοθεραπεία και επιχειρησιακή συνέχεια (business continuity).

Βάσει της μελέτης των Sharma et al. [6], αναδεικνύεται η ολοένα αυξανόμενη σημασία της χρήσης IoT τεχνολογιών στον τομέα της γεωργίας ακριβείας, όπου η αξιοποίηση αισθητήρων για την παρακολούθηση παραμέτρων όπως η υγρασία, η θερμοκρασία, η αγωγιμότητα και τα επίπεδα θρεπτικών συστατικών του εδάφους (Αζώτου, Φωσφόρου, Καλίου), επιτρέπει την ακριβή λήψη αποφάσεων σε πραγματικό χρόνο.

Η προσέγγιση αυτή ευθυγραμμίζεται με την ανάγκη για ευέλικτες, αποκεντρωμένες υποδομές λήψης αποφάσεων, οι οποίες υποστηρίζονται από μηχανισμούς edge analytics και cloud ενορχήστρωσης. Στο πλαίσιο

αυτό, η μελέτη των Akhtar et al. [7] τονίζει τη σημασία της ενσωμάτωσης του edge computing για την επεξεργασία δεδομένων από αισθητήρες σε πραγματικό χρόνο, επιτρέποντας την αξιολόγηση του εδάφους και την παρακολούθηση ρύπων με μεγαλύτερη αποτελεσματικότητα. Υποστηρίζει, επιπλέον, πως η ενσωμάτωση τεχνικών μηχανικής μάθησης στην αλυσίδα επεξεργασίας των δεδομένων, όπως η αυτόματη πρόβλεψη της καταλληλότητας του εδάφους για συγκεκριμένες καλλιέργειες, ενισχύει την αξία των IoT συστημάτων και απαιτεί αρχιτεκτονική σχεδίαση που υποστηρίζει δυναμική ανάλυση και διαλειτουργικότητα.

Παρά το γεγονός ότι η τεχνολογία δεν έχει ακόμη ενσωματωθεί πλήρως στον αγροτικό τομέα, παρατηρείται αυξανόμενη τάση υιοθέτησης καινοτόμων λύσεων, ιδιαίτερα στο επίπεδο του αγρού. Μια από τις πιο προσιτές προσεγγίσεις βασίζεται στην αξιοποίηση αισθητήρων (θερμοκρασίας, υγρασίας, φωτός, βροχόπτωσης, ταχύτητας ανέμου) οι οποίοι σχηματίζουν ένα τοπικό δίκτυο συλλογής δεδομένων στον χώρο της καλλιέργιας. Τα δεδομένα που συλλέγονται μεταφέρονται σταδιακά προς υπολογιστικές μονάδες αυξημένης ισχύος, σχηματίζοντας μια πολυεπίπεδη, αποκεντρωμένη υποδομή που επιτρέπει τόσο την τοπική επεξεργασία όσο και την κεντρική αποθήκευση και ανάλυση [8].

Τα ευρήματα αυτά ενισχύουν την άποψη ότι η αρχιτεκτονική ενός ανθεκτικού IoT συστήματος στον αγροδιατροφικό τομέα πρέπει να υποστηρίζει συνεχές monitoring, on-device προεπεξεργασία και ασφαλή, επεκτάσιμη μετάδοση δεδομένων, προκειμένου να επιτευχθεί πλήρης αυτοματοποίηση και ευστάθεια λειτουργίας σε ετερογενή, διασυνδεδεμένα περιβάλλοντα πεδίου.

Με στόχο, λοιπόν, την αρχιτεκτονική ενίσχυση του συστήματος *Nostradamus* προτείνεται, ως απόρροια των παραπάνω μελετών, η ενσωμάτωση μηχανισμών caching, observability και self-healing. Παρότι υπάρχουν επιμέρους τεχνολογίες που προσφέρουν τέτοιες δυνατότητες, η υιοθέτησή τους σε πραγματικές πολυεπίπεδες IoT πλατφόρμες συνοδεύεται από σημαντικές προκλήσεις. Στα κατανεμημένα συστήματα, η προσθήκη μιας νέας υπηρεσίας ή λειτουργικότητας συνοδεύεται συχνά από μη προβλέψιμες επιπτώσεις στην απόδοση του συστήματος, όπως αναφέρει επανειλημμένα ο Kleppmann [9]. Αν και οι μηχανισμοί όπως το caching και το observability αποσκοπούν στην βελτίωση της εμπειρίας και της διαχειρισιμότητας, μπορούν, ως παράπλευρη συνέπεια, να οδηγήσουν σε αύξηση του latency ή της κατανάλωσης πόρων. Το φαινόμενο αυτό αποδίδεται στην πολυπλοκότητα των αλληλεπιδράσεων μεταξύ μικροϋπηρεσιών και την έλλειψη πλήρους ορατότητας σε χαμηλό επίπεδο.

Κατ' επέκταση, η ερευνητική συνεισφορά εστιάζει όχι μόνο στην επι-

λογή σχετικών τεχνολογιών, αλλά κυρίως στη συνεκτική και δυναμική ενορχήστρωσή τους, με στόχο την επίτευξη αυτονομίας και διαχειρισιμότητας σε cloud-native κατανεμημένα περιβάλλοντα υψηλής πολυπλοκότητας, για την υποστήριξη αυτού του συστήματος πραγματικού χρόνου στο χώρο του *food security*.

3
Θεωρητικό Υπόβαθρο

Η εξέλιξη των Κυβερνο-Φυσικών Συστημάτων (CPS) και των εφαρμογών IoT βασίζεται σε θεμελιώδεις αρχιτεκτονικές αρχές που συνδυάζουν την αποδοτική διαχείριση της φυσικής πληροφορίας με τις δυνατότητες της σύγχρονης υπολογιστικής τεχνολογίας. Σε αυτό το κεφάλαιο παρουσιάζονται τα βασικά αρχιτεκτονικά μοντέλα, οι μέθοδοι διασύνδεσης υποσυστημάτων και τα πρότυπα διαχείρισης δεδομένων που επικρατούν στα σύγχρονα CPS.

3.1 Αρχιτεκτονικες CPS και Κατανεμημενα Συστηματα

Τα CPS δομούνται τυπικά σε τρία επίπεδα: το **φυσικό επίπεδο** (sensing/actuation), το **επίπεδο δικτύου** (networking/communication) και το **κυβερνητικό ή υπολογιστικό επίπεδο** (cyber/computation). Το φυσικό επίπεδο περιλαμβάνει αισθητήρες και ενεργοποιητές, το δίκτυο μεταφέρει τα δεδομένα, ενώ το κυβερνητικό επίπεδο διαχειρίζεται την επεξεργασία, την ανάλυση και τη λήψη αποφάσεων.

Η σύγχρονη σχεδίαση CPS ακολουθεί τα παρακάτω αρχιτεκτονικά μοτίβα:

- **Layered (στρωματοποιημένη) αρχιτεκτονική**: Διαχωρισμός φυσικών, δικτυακών και υπολογιστικών λειτουργιών, διευκολύνοντας

τη διαλειτουργικότητα και τη συντήρηση.

- **Κατανεμημένη επεξεργασία**: Τα δεδομένα δεν συγκεντρώνονται υποχρεωτικά σε έναν κεντρικό κόμβο, αλλά επεξεργάζονται τοπικά (*edge/fog computing*) ή υβριδικά, βελτιώνοντας την καθυστέρηση και την ανθεκτικότητα.
- **Event-driven και streaming αρχιτεκτονική**: Αντί της παραδοσιακής batch επεξεργασίας, τα δεδομένα ρέουν ως ακολουθίες γεγονότων (streams), επιτρέποντας την άμεση αντίδραση σε αλλαγές του περιβάλλοντος.
- **Μικροϋπηρεσίες (Microservices)**: Ανάπτυξη του λογισμικού σε μικρές, αυτόνομες υπηρεσίες με σαφή όρια ευθύνης και ανεξάρτητο κύκλο ζωής.
- **Loose coupling & αναγνωσιμότητα**: Η επικοινωνία μεταξύ των υποσυστημάτων βασίζεται σε ασύγχρονα μηνύματα (π.χ. μέσω pub/sub patterns), διατηρώντας την αυτονομία και διευκολύνοντας την κλιμάκωση.

3.2 Streams, Event-Driven και Batch vs Real-Time Processing

Η *ροή δεδομένων* (stream) ορίζεται ως η συνεχής αλληλουχία δεδομένων που παράγονται από αισθητήρες ή άλλες πηγές και διακινούνται ασύγχρονα εντός του συστήματος. Η **event-driven αρχιτεκτονική** επιτρέπει τη λήψη αποφάσεων ή την ενεργοποίηση ενεργειών αμέσως με την εμφάνιση ενός γεγονότος, κάτι που είναι κρίσιμο για εφαρμογές με απαιτήσεις απόκρισης σε πραγματικό χρόνο, όπως η γεωργία ακριβείας, το predictive maintenance, ή η αυτονομία ρομποτικών συστημάτων.

Σε αντίθεση, τα παραδοσιακά *batch systems* επεξεργάζονται δεδομένα συγκεντρωτικά και περιοδικά, οδηγώντας σε υστέρηση που δεν είναι αποδεκτή για CPS με κρίσιμες λειτουργίες. Η υιοθέτηση stream processing (με αρχιτεκτονικές Lambda/Kappa) προσφέρει καλύτερη προσαρμοστικότητα και αυτονομία, ενώ μειώνει την καθυστέρηση λήψης αποφάσεων.

3.3 Αρχιτεκτονικεσ Streams: Lambda και Kappa

Οι δύο επικρατέστερες αρχιτεκτονικές για επεξεργασία ροών είναι:

- **Lambda**: Συνδυάζει speed layer για άμεση επεξεργασία (με πιθανό μικρότερο accuracy) και batch layer για πλήρη ανάλυση με υψηλότερη ακρίβεια, υποστηρίζοντας τόσο real-time όσο και offline analytics.
- **Kappa**: Ενιαία προσέγγιση, όπου όλα τα δεδομένα αντιμετωπίζονται ως streams, απλοποιώντας τη διαχείριση και επανεπεξεργασία γεγονότων χωρίς ανάγκη ξεχωριστού batch υποσυστήματος.

Η επιλογή μοντέλου εξαρτάται από το αν προτεραιότητα δίνεται στην αμεσότητα ή στην αναλυτική επεξεργασία και από τη συνολική πολυπλοκότητα συντήρησης. Σε πρακτικό επίπεδο, υλοποιήσεις όπως το *Apache Kafka* (messaging/stream log), *Apache Flink* και *Apache Spark* (stream processing) έχουν επικρατήσει ως τεχνολογικά standards, λόγω της ευελιξίας, αξιοπιστίας και δυνατότητας διασύνδεσης με άλλα υποσυστήματα.

3.4 Διασυνδεση Υποσυστηματων

Για την αλληλεπίδραση πολλαπλών, ετερογενών υποσυστημάτων, υιοθετείται το **publish/subscribe** μοτίβο. Αυτό επιτρέπει την ασύγχρονη και χαλαρά συνδεδεμένη (*loosely coupled*) επικοινωνία μεταξύ παραγωγών (*publishers*) και καταναλωτών (*subscribers*), διασφαλίζοντας την αυτονομία, την επεκτασιμότητα και τη δυνατότητα αλλαγών στην τοπολογία του συστήματος χωρίς κεντρικό συντονισμό.

Η ποιότητα υπηρεσίας (*Quality of Service, QoS*) αποκτά ιδιαίτερη σημασία, καθώς καθορίζει το επίπεδο αξιοπιστίας στη διανομή μηνυμάτων, ειδικά σε περιβάλλοντα με ασταθή συνδεσιμότητα (π.χ. αγροτικά δίκτυα, βιομηχανικά IoT). Για ποιοτικά περιορισμένες συσκευές και low-power δίκτυα, πρωτόκολλα όπως το *MQTT* προσφέρουν ελαφριά, αξιόπιστη και ασφαλή ανταλλαγή μηνυμάτων, ενισχύοντας την ανθεκτικότητα του συστήματος.

3.5 Αποθηκευση, Consistency Models και Caching

Η αποθήκευση δεδομένων σε CPS συνιστά αρχιτεκτονική πρόκληση λόγω του όγκου, της ταχύτητας παραγωγής και της ανάγκης για real-

time προσπέλαση. Κατανεμημένα συστήματα βάσεων (NoSQL, wide-column stores) με replication και partitioning διασφαλίζουν διαθεσιμότητα και fault tolerance.

Η επιλογή consistency model (strong, eventual, causal) εξαρτάται από τις απαιτήσεις της εφαρμογής για ακρίβεια vs ταχύτητα και διαθεσιμότητα. Σε συστήματα που μπορούν να αντέξουν μικρές απώλειες ή προσωρινή ασυνέπεια, το eventual consistency αποτελεί πρακτικό συμβιβασμό.

Επισημαίνεται επίσης, ότι η χρήση *in-memory caching* (π.χ. με Redis/Memcached) μειώνει δραστικά το latency για δεδομένα που προσπελαύνονται συχνά ή απαιτούν άμεση διαθεσιμότητα, με trade-offs σε αντοχή σε αποτυχίες (durability).

3.6 Διαχειριση Υποδομης και Cloud-Native Patterns

Η ανάπτυξη CPS σε cloud ή υβριδικά περιβάλλοντα απαιτεί αυτοματοποιημένη διαχείριση υποδομής και υπηρεσιών. Εδώ κυριαρχεί η χρήση πλατφόρμων όπως το Kubernetes, που υλοποιούν:

- **Δηλωτική διαχείριση (declarative infrastructure)**: Η επιθυμητή κατάσταση του συστήματος ορίζεται μέσω configuration, και ο orchestrator διασφαλίζει ότι αυτή τηρείται.
- **Αυτόματη κλιμάκωση (auto-scaling)**, **αυτοΐαση (self-healing)**: Προσθήκη/αφαίρεση pods, επανεκκινήσεις σε περίπτωση αποτυχιών, health checks.
- **Παρατηρησιμότητα (observability)**: Συλλογή μετρικών, logs, tracing και alerting, με ενδεικτικά εργαλεία Prometheus, Grafana, ELK/EFK stack.
- **Διαμοιρασμός πόρων (virtualization)**: Ευνοϊκή μεθοδολογία για την απομόνωση εφαρμογών και την καλύτερη αξιοποίηση του διαθέσιμου hardware μέσω εικονικών μηχανών ή containers.
- **Αυτοματοποιημένος έλεγχος με Kubernetes Operators**: Επέκταση του control plane του Kubernetes, με αρχές αυτοματοποιημένων συστημάτων ελέγχου για τη διαχείριση σύνθετων, κατανεμημένων εφαρμογών.

- **Service discovery, load balancing και secrets management**: Απαραίτητα για τη διασύνδεση μικροϋπηρεσιών και τη διατήρηση ασφάλειας και διαθεσιμότητας.

Έτσι, η υποδομή μεταμορφώνεται σε έναν ζωντανό οργανισμό που προσαρμόζεται διαρκώς στις απαιτήσεις του συστήματος και του περιβάλλοντος, προσφέροντας ευελιξία και ανθεκτικότητα χωρίς ανθρώπινη παρέμβαση. Με αυτόν τον τρόπο, περιορίζονται τα λάθη του ανθρώπινου παράγοντα, καθώς κρίσιμες λειτουργίες αυτοματοποιούνται και αναλαμβάνονται από το ίδιο το σύστημα.

3.7 Συμπερασματα Αρχιτεκτονικης

Η επιτυχία ενός σύγχρονου CPS/IoT συστήματος εξαρτάται από την υιοθέτηση δοκιμασμένων αρχιτεκτονικών προτύπων, τη σαφή οριοθέτηση των στρωμάτων (sensing, networking, processing, storage, observability), και τη χρήση τεχνολογιών που υποστηρίζουν real-time streaming, fault tolerance, scalability, automation και ασφάλεια. Οι επιλογές σε επίπεδο εργαλείων (Kafka, Flink, Cassandra, Redis, Kubernetes, Prometheus, κτλ.) υλοποιούν στην πράξη τα θεμελιώδη design patterns που περιγράφηκαν σε αυτό το κεφάλαιο, αποτελώντας τη "γέφυρα" μεταξύ θεωρίας και εφαρμογής.

4
Εργαλεία

Τα Κυβερνο-Φυσικά Συστήματα (CPS) είναι συστήματα που αποτελούνται από ένα φυσικό στοιχείο το οποίο ελέγχεται ή παρακολουθείται από ένα κυβερνητικό (cyber) στοιχείο, έναν αλγόριθμο βασισμένο σε υπολογιστή. Με στόχο να μετασχηματίσουν τον τρόπο με τον οποίο οι άνθρωποι αλληλεπιδρούν με τα μηχανικά συστήματα, τα νέα έξυπνα CPS οδηγούν την καινοτομία σε διάφορους τομείς, βασικός εκ των οποίων αποτελεί η γεωργία [10]. Η αρχιτεκτονική των CPS βασίζεται σε τρία βασικά επίπεδα: το φυσικό επίπεδο (physical layer), όπου καταγράφονται και παράγονται τα δεδομένα μέσω αισθητήρων· το επίπεδο δικτύου (network layer), που εξασφαλίζει τη μεταφορά των δεδομένων· και το κυβερνητικό ή υπολογιστικό επίπεδο (cyber layer), όπου λαμβάνονται αποφάσεις βάσει των εισερχόμενων δεδομένων. Η συνύπαρξη αυτών των στρωμάτων σε ένα κοινό σύστημα καθιστά τα CPS ιδιαίτερα κατάλληλα για εφαρμογές που απαιτούν χαμηλή καθυστέρηση, αξιοπιστία και αυτονομία. Όπως προαναφέρθηκε, στο πεδίο της γεωργίας ακριβείας, τα CPS διαδραματίζουν καθοριστικό ρόλο, καθώς συνδυάζουν αισθητήρες πεδίου, μηχανισμούς ελέγχου άρδευσης, και αλγορίθμους πρόβλεψης βασισμένους σε δεδομένα για να εξασφαλίσουν βέλτιστες συνθήκες καλλιέργειας. Οι τεχνολογίες αυτές επιτρέπουν τη δυναμική λήψη αποφάσεων, μειώνουν τις απώλειες και αυξάνουν την αποδοτικότητα σε όλα τα στάδια της παραγωγής.

Για να επιτευχθεί όμως η πλήρης δυναμική των CPS, απαιτούνται ισχυρές υποδομές διασύνδεσης και διαχείρισης δεδομένων. Εδώ εντάσ-

σεται η ανάγκη για αξιόπιστες messaging πλατφόρμες, όπως το Apache Kafka, που διασφαλίζουν την συνεχή και αξιόπιστη ροή πληροφοριών ανάμεσα στα υποσυστήματα ενός CPS. Το Apache Kafka αποτελεί ένα - πλέον - de facto πρότυπο για την υλοποίηση τέτοιων messaging υποδομών, χάρη στη δυνατότητα του να αποθηκεύει τα μηνύματα σε μορφή καταγραφής (log-based) [11]. Η αρχιτεκτονική του Kafka ενδείκνυται ιδιαίτερα για περιβάλλοντα που απαιτούν real ή near-real time επεξεργασία γεγονότων, καθώς παρέχει υψηλή διαθεσιμότητα, εγγυημένη διανομή μηνυμάτων και δυνατότητα οριζόντιας κλιμάκωσης. Συγκριτικές μελέτες [12] επιβεβαιώνουν ότι το Kafka παρουσιάζει σημαντικό πλεονέκτημα σε όρους throughput και fault tolerance σε σχέση με άλλες προσεγγίσεις, γεγονός που το καθιστά κατάλληλο για απαιτητικές IoT εφαρμογές με αυξημένο όγκο και ταχύτητα δεδομένων.

Σε ένα σύστημα όπως το *Nostradamus*, όπου η ροή των δεδομένων είναι αδιάκοπη και εξελίσσεται σε πραγματικό χρόνο, η ταχεία εισαγωγή και εξαγωγή των δεδομένων εκτιμάται, τόσο για λόγους απόδοσης όσο και για την ελαχιστοποίηση της κατανάλωσης υπολογιστικών πόρων. Δοθέντος ενός συνόλου αισθητήρων τοποθετημένων σε έναν αγρό, οι οποίοι παράγουν συνεχώς δεδομένα, ο κεντρικός broker πρέπει να τα λαμβάνει ορθά και εντός λογικών χρονικών πλαισίων, ενώ ταυτόχρονα να τα επεξεργάζεται χωρίς να καταπονεί το συνολικό σύστημα. Συνεπώς, πρέπει να ληφθούν υπόψη τόσο η ποιότητα υπηρεσίας (Quality of Service - QoS) της messaging υποδομής όσο και οι εγγυήσεις παράδοσης που αυτή προσφέρει. Στη συγκεκριμένη εφαρμογή, η απώλεια ενός μεμονωμένου δείγματος αισθητήρα δεν επηρεάζει σημαντικά τη συνολική ανάλυση, επομένως η πολιτική "**at-least-once**" αποτελεί μια ασφαλή και επαρκή επιλογή. Οι Kreps et al. [13] προσθέτουν, πως το Kafka σχεδιάστηκε εξαρχής με γνώμονα το υψηλό throughput, αποφεύγοντας περίπλοκους μηχανισμούς όπως το two-phase commit και υιοθετώντας πιο αποδοτικές λύσεις για περιπτώσεις όπου η απώλεια μηνυμάτων είναι αποδεκτή.

Τα δεδομένα που εισάγονται στα topics του *Kafka* επεξεργάζονται και καταλήγουν είτε σε επόμενα topics για μετέπειτα ανάλυση, είτε απευθείας σε αποθηκευτικά συστήματα. Ο τρόπος με τον οποίο πραγματοποιείται η εν λόγω επεξεργασία οφείλει να είναι, εν γένει, σε πραγματικό ή σχεδόν πραγματικό χρόνο, καθώς τα δεδομένα παράγονται και εισάγονται στο σύστημα σε συνθήκες online ροής. Επομένως, εργαλεία όπως το *Apache Flink*, τα οποία παρέχουν native υποστήριξη για stream processing με χαμηλή καθυστέρηση, καθίστανται ιδανικά για την υλοποίηση του ενδιάμεσου επιπέδου επεξεργασίας.

Το *Flink* επιτρέπει τον ορισμό παραθύρων (windows) με βάση τον

χρόνο γεγονότος (event time), την υλοποίηση πολύπλοκων λειτουργιών (όπως filtering, aggregation, enrichment), καθώς και την ενσωμάτωση με messaging και αποθηκευτικά συστήματα, μεταξύ των οποίων και το *Apache Kafka* και το *Apache Cassandra*. Επιπλέον, μέσω του μηχανισμού state management που διαθέτει, διασφαλίζεται η αξιοπιστία της επεξεργασίας, ακόμη και σε περιπτώσεις προσωρινών αποτυχιών. Η ενσωμάτωση του σε αρχιτεκτονικές τύπου CPS επιτρέπει τη δημιουργία πραγματικά αντιδραστικών συστημάτων, τα οποία μπορούν να λαμβάνουν αποφάσεις βάσει εξελισσόμενων δεδομένων, χωρίς να απαιτείται off-line επεξεργασία ή χρονική υστέρηση.

Η αποθήκευση μεγάλου όγκου δεδομένων σε κατανεμημένα συστήματα βασίζεται - κυρίως - σε βάσεις δεδομένων τύπου *wide-column*, με το *Apache Cassandra* να αποτελεί ένα από τα πιο διαδεδομένα και ώριμα συστήματα σε αυτόν τον χώρο. Το Cassandra ακολουθεί το μοντέλο *eventual consistency*, υποστηρίζει κατανεμημένη αποθήκευση με replication και partitioning, και έχει σχεδιαστεί για *write-heavy* εφαρμογές [14]. Σε περιβάλλοντα IoT, όπου οι συσκευές παράγουν συνεχώς δεδομένα τηλεμετρίας (π.χ. θερμοκρασία, τάση, ρεύμα) με υψηλή συχνότητα, η Cassandra μπορεί να λειτουργήσει ως backend αποθήκευσης για ροές δεδομένων σχεδόν σε πραγματικό χρόνο, διασφαλίζοντας υψηλή διαθεσιμότητα και συνεχή εγγραφή με χαμηλό latency. Η προσέγγιση αυτή έχει εφαρμοστεί επιτυχώς σε σενάρια όπως η παρακολούθηση φωτοβολταϊκών μονάδων μέσω Raspberry Pi [15], όπου το σύστημα συλλέγει και αποθηκεύει μετρήσεις αισθητήρων κάθε 15 λεπτά για περαιτέρω ανάλυση και βελτιστοποίηση απόδοσης. Αντίστοιχες αρχιτεκτονικές υποδεικνύουν τη σημασία της επιλογής αποθηκευτικού συστήματος που να ανταποκρίνεται τόσο σε επιχειρησιακές ανάγκες χαμηλής καθυστέρησης όσο και σε απαιτήσεις αξιοπιστίας και επεκτασιμότητας.

Στη συγκεκριμένη περίπτωση, η υψηλή απόδοση (high performance) της *Cassandra* σε συνδυασμό με την εύκολη και ελαστική επεκτασιμότητα (elastic scalability), καθιστά το σύστημα ιδανικό για εφαρμογές πραγματικού χρόνου με έντονη ροή δεδομένων. Η δυνατότητα προσθήκης ή αφαίρεσης κόμβων (nodes) χωρίς διακοπή λειτουργίας επιτρέπει την ομαλή προσαρμογή στις αυξομειώσεις φορτίου, επιτρέποντας το self-healing, διατηρώντας παράλληλα σταθερή τη χρονική απόκριση. Καθώς το σύστημα απαιτεί ταχεία επεξεργασία, άμεση καταχώρηση και αξιόπιστη αποθήκευση δεδομένων αισθητήρων πεδίου, η επιλογή μιας αρχιτεκτονικής βασισμένης στην *Cassandra* εξασφαλίζει υψηλή διαθεσιμότητα και ανθεκτικότητα σε αποτυχία. Το λειτουργικό όφελος που προκύπτει από αυτήν τη σχεδίαση δεν είναι απλώς επιθυμητό αλλά κρίσιμης σημασίας, ιδιαίτερα σε περιβάλλοντα με απαιτήσεις χαμηλού

latency, συνεχούς διαθεσιμότητας και γραμμικής επεκτασιμότητας.

Τελικός στόχος του συστήματος Nostradamus είναι η αξιοποίηση των αποθηκευμένων και επεξεργασμένων δεδομένων σε εφαρμογές διεπαφής, ώστε οι χρήστες να μπορούν να λαμβάνουν οπτικοποιημένη και άμεσα αξιοποιήσιμη πληροφόρηση σχετικά με την καλλιέργειά τους. Οι κόμβοι της βάσης δεδομένων διαδραματίζουν κρίσιμο ρόλο στην τροφοδότηση των εφαρμογών με δεδομένα σε πραγματικό χρόνο. Η ανάγκη για άμεση προσπέλαση σε δεδομένα, ιδιαίτερα σε περιβάλλοντα όπου η εισροή πληροφορίας είναι συνεχής και εν δυνάμει μαζική, καθιστά απαραίτητη τη χρήση μηχανισμών προσωρινής αποθήκευσης (in-memory caching).

Στο πλαίσιο αυτό, τεχνολογίες όπως το *Memcached* [16] και το *Redis* [17] προσφέρουν σημαντικά πλεονεκτήματα, μέσω της υλοποίησης γρήγορων και αποδοτικών μηχανισμών αποθήκευσης ζευγών κλειδιού-τιμής (key-value pairs). Οι εν λόγω λύσεις συμβάλλουν ουσιαστικά στη μείωση του χρόνου και του κόστους προσπέλασης σε δεδομένα που διαφορετικά θα απαιτούσαν αναζήτηση στη βάση. Η βασική τους διαφοροποίηση εντοπίζεται κυρίως στο αρχιτεκτονικό τους μοντέλο και στον τρόπο διαχείρισης της ταυτόχρονης εκτέλεσης, με το *Memcached* να βασίζεται σε multithreaded επεξεργασία, ενώ το *Redis* αξιοποιεί μοντέλο event loop.

Η ανάγκη για παρατηρησιμότητα, αποδοτική διαχείριση προσωρινών δεδομένων και σταθερή ροή μηνυμάτων σε περιβάλλοντα με κατανεμημένη υπολογιστική λογική, καθιστά την πλατφόρμα *Kubernetes*, συχνά, απαραίτητο δομικό στοιχείο. Η δυνατότητα αυτόματης επανεκκίνησης αποτυχημένω πόρων (self-healing), η υποστήριξη οριζόντιας κλιμάκωσης (horizontal pod autoscaling) και η ενσωματωμένη παρακολούθηση της κατάστασης των υπηρεσιών (liveness/readiness probes), προσδίδουν λειτουργική ανθεκτικότητα και διατηρούν τη διαθεσιμότητα του συστήματος σε υψηλά επίπεδα, ακόμη και σε συνθήκες έντονου φορτίου ή υλικών αποτυχιών.

Πέραν της αυτοματοποιημένης διαχείρισης πόρων, το *Kubernetes* παρέχει μια ενιαία πλατφόρμα παρατηρησιμότητας. Μέσω της ενσωμάτωσης εργαλείων όπως το *Prometheus* για συλλογή μετρικών, το *Grafana* για οπτικοποίηση και το *AlertManager* για την αποστολή ειδοποιήσεων, [18] - κοινώς τη **lingua franca** του *observability* - ενισχύεται η κατανόηση της δυναμικής συμπεριφοράς του συστήματος σε πραγματικό χρόνο. Η εγγενής υποστήριξη μηχανισμών για service discovery, load balancing και declarative configuration, επιτρέπει την ευέλικτη ανάπτυξη και τον συντονισμό μικροϋπηρεσιών, διευκολύνοντας την επίτευξη στόχων υψηλής διαθεσιμότητας και επεκτασιμότητας σε cloud-native περιβάλλοντα. Συνεπώς, υιοθετώντας αντίστοιχα πρότυπα, μπορούν να προληφθούν

ανεπιθύμητες καταστάσεις και να διασφαλιστεί η διατήρηση της ομαλής λειτουργίας ακόμα και υπο συνθήκες υψηλής πολυπλοκότητας.

5
Υλοποιήσεις

6
Πειράματα

7
Συμπεράσματα

Βιβλιογραφία

[1] Denise Ratasich, Faiq Khalid, Florian Geissler, Radu Grosu, Muhammad Shafique, and Ezio Bartocci. A roadmap toward the resilient internet of things for cyber-physical systems. *IEEE Access*, 2019.

[2] Godson Michael D'silva, Azharuddin Khan, Gaurav, and Siddhesh Bari. Real-time processing of iot events with historic data using apache kafka and apache spark with dashing framework. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2017.

[3] Constantinos Styliaras, Dimitrios Kyrkilis, and Symeon Semasis. *The Role of Agriculture in Economic Growth in Greece*. 2013.

[4] Anthony Callo and Mo Mansouri. Food security in global food distribution networks: A systems thinking approach. In *2024 IEEE International Systems Conference (SysCon)*, 2024.

[5] Hwiwon Lee, Sosun Kim, and Huy Kang Kim. Sok: Demystifying cyber resilience quantification in cyber-physical systems. In *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022.

[6] Dr. Rashmi Sharma, Vishal Mishra, and Suryansh Srivastava. Enhancing crop yields through iot-enabled precision agriculture. In *2023 International Conference on Disruptive Technologies (ICDT)*, 2023.

[7] Mohammad Nishat Akhtar, Abdurrahman Javid Shaikh, Ambareen Khan, Habib Awais, Elmi Abu Bakar, and Abdul Rahim Othman. Smart sensing with edge computing in precision agriculture for soil assessment and heavy metal monitoring: A review. *Agriculture*, "2021".

[8] Hanzhe Li, Xiangxiang Wang, Yuan Feng, Yaqian Qi, and Jingxiao Tian. Driving intelligent iot monitoring and control through cloud computing and machine learning, 2024. URL *https://arxiv.org/abs/2403.18100*.

[9] Martin Kleppmann. *Designing Data-Intensive Applications*. O'Reilly Media, 2017.

[10] Mario Lezoche and Hervé Panetto. Cyber-physical systems, a new formal paradigm to model redundancy and resiliency. 2018. URL *http://arxiv.org/abs/1810.06911*.

[11] Rishika Shree, Tanupriya Choudhury, Subhash Chand Gupta, and Praveen Kumar. Kafka: The modern platform for data management and analysis in big data domain. In *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, 2017.

[12] Chung Lai Deryck Ho, Chung–Horng Lung, and Zhengding Mao. Comparative analysis of real-time data processing architectures: Kafka versus mqtt broker in iot. In *2024 IEEE 4th International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, 2024.

[13] Neha Narkhede Jay Kreps and Jun Rao. Kafka: a distributed messaging system for log processing. In *Proceedings of the NetDB Workshop*, 2011. URL *https://api.semanticscholar.org/CorpusID:18534081*.

[14] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 2010.

[15] Arbër Sh. Perçuku, Daniela V. Minkovska, Lyudmila Y. Stoyanova, and Arta E. Abdullahu. Iot using raspberry pi and apache cassandra on pv solar system. In *2020 XXIX International Scientific Conference Electronics (ET)*, 2020.

[16] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. Scaling memcache at facebook. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association, 2013.

[17] Josiah Carlson. *Redis in Action*. Manning Publications, 2013.

[18] Sneha M. Pragathi B.C., Hrithik Maddirala. Implementing an effective infrastructure monitoring solution with prometheus and grafana. *International Journal of Computer Applications*, 2024.